MiWi™

MiWi™ Migration Guide

Introduction

This guide provides information for migrating MiWi applications implemented on:

- MiWi v6.3 available for SAM platforms (SAMR21 and SAMR30) to MiWi v6.4
- MiWi v6.2 available for SAM platforms (SAMR21 and SAMR30) to MiWi v6.3
- MiWi v6.1 available for SAM platforms (SAMR21 and SAMR30) to MiWi v6.2
- MiWi v6.0 available for SAM platforms (SAMR21 and SAMR30) to MiWi v6.1
- MiWi protocol available in Microchip Libraries for Applications (MLA) to MiWi v6.0 (SAM platforms - SAMR21and SAMR30)

Table of Contents

1. Migrating MiWi v6.3 to v6.4

This section provides the required information to migrate MiWi applications implemented on MiWi v6.3 available in the SAM platforms (SAMR21 and SAMR30) to MiWi v6.4.

1.1 New Features and Enhancements of MiWi v6.4

- WiDBG - Addition of Wireless Debug Support for Mesh. For more information, refer to the MiWi Quick Start Guide
- Bug fixes/enhancements to the previous versions of MiWi Stack.

1.1.1 Enhancements

- Version information of the MiWi stack is defined in `miwi_api.h` file.

1.2 File Additions

The following are the new files added to the MiWi v6.4 stack:

1. Supporting files for OTAU Debug are available in ASF at `\thirdparty\wireless\miwi\services\otau\debug`.
2. Generic circular buffer for Debug are available in ASF at `\thirdparty\wireless\miwi\services\otau`.

1.3 API Changes

The following table lists the API changes with respect to the old APIs.

Table 1-1. API Changes for Old APIs

S. No.	Old APIs (v6.3)	New APIs (v6.4)	Comments
1	`void PHY_DataReq(uint8_t *data)`	`void PHY_DataReq(PHY_DataReq_t* phyDataReq)`	This is to handle the possibility of calling this API from MiMAC and also directly from the OTAU module.

2. Migrating MiWi v6.2 to v6.3

This section provides the required information to migrate MiWi applications implemented on MiWi v6.2 available in the SAM platforms (SAMR21 and SAMR30) to MiWi v6.3.

2.1 New Features and Enhancements of MiWi v6.3

- Architecture redesign of MiWi P2P and Star protocol source code
- Enhancements to the stack with respect to design and performance
- Sleep improvements
- Bug fixes

2.1.1 Architecture Redesign of MiWi P2P and Star Protocol Source Code

The following points are addressed in the redesigned source code:

- Blocking calls are removed by implementing APIs with callback function.
- Timer callbacks are implemented using a common software timer instead of `MiWi_TickGet`.
- Transmit messages are queued and sent sequentially.
- In `MiApp_ProtocolInit`, a new configurable option is provided to boot from persistent memory.
- New API `MiApp_SubscribeReConnectionCallback` is added to know successful reconnection. This callback is called upon successful restoration of existing network parameters upon `MiApp_ProtocolInit`.
- New API `MiApp_ReadyToSleep` is added to check the P2P/Star stack and whether it allows sleep and how much time it allows to sleep for proper stack operations.
- Outgoing data and data waiting for acknowledgment are queued.
- Packet forward and acknowledgment (if configured) in Star protocol is implemented as part of Star stack itself.
- New API `MiApp_SubscribeLinkFailureCallback` is implemented to subscribe a function to be notified upon connection lost.
- Long address is stored in persistent memory which helps to use the same long address which was used before, when power is lost and PDS is used for network parameter restore.

Note: The P2P and Star protocol is not modified; only the code is redesigned to support non-blocking message transfer.

For more information, refer to the MiWi Software Design Guide.

2.1.2 Sleep Mode Enhancements

The Sleep feature is improved in MiWi P2P, Star and Mesh protocols.

The following table shows the power values of Sleep mode for the SAMR30 and the SAMR31 devices.

Table 2-1. Power Values

Sleep Mode	SAMR30	SAMR21
Standby mode	1.5 uA	25 uA
Backup mode	0.7 uA	Not applicable

2.2 File Changes

The following are the modification to files in MiWi v6.3 stack:

- `\thirdparty\wireless\miwi\source\miwi_p2p_star\miwi_p2p.c/.h` files are removed.
- `\thirdparty\wireless\miwi\source\miwi_p2p_star\miwi_p2p_star.c/.h` files are added.

2.3 MiApp API Changes

The following table lists the API changes with respect to the old APIs.

Table 2-2. API Changes for Old APIs

S. No.	Old APIs (v6.2)	New APIs (v6.3)
1	Not available for P2P and Star protocol	`bool MiApp_SubscribeReConnectionCallback (ReconnectionCallback_t callback);`
2	Not available for P2P and Star protocol	`bool MiApp_SubscribeLinkFailureCallback(LinkFailureCallback_t callback);`
3	Not available for P2P and Star protocol	`bool MiApp_SubscribeReConnectionCallback(ReconnectionCallback_ callback);`
4	Not Available for P2P and Star protocol	`bool MiApp_ReadyToSleep(uint32_t* sleepTime);`
5	`void MiApp_BroadcastConnectionTable(void)`	Removed
6	`bool SW_Ack_SrED(uint8_t *)`	Removed
7	`void send_link_status(void);`	Removed
8	`bool MiApp_ResyncConnection(uint8_t ConnectionIndex, uint32_t ChannelMap);`	`bool MiApp_ResyncConnection(uint8_t ConnectionIndex, uint32_t ChannelMap,resyncConnection_callback_t callback)`
9	`void Find_InActiveDevices(void)`	Removed
10	`void MiApp_leave_network(void)`	Removed
11	`bool MiApp_UnicastStar (bool SecEn)`	Removed

The parameter `defaultParametersRomOrRam_t` of `MiApp_ProtocolInit` is changed similar to Mesh protocol where the application can provide memory for the connection table, active scan results, additional node ID and `networkFreezerRestore` instructs the stack to restore the network freezer based on application input.

```
typedef struct __defaultParametersRomOrRam
{
           /* Connection Table Memory Pointer */
           CONNECTION_ENTRY  *ConnectionTable;
           /* Active Scan Results Memory Pointer */
           ACTIVE_SCAN_RESULT  *ActiveScanResults;
           /* Additional Node ID Memory Pointer */
           uint8_t *AdditionalNodeID;
           /* Set option to restore from network freezer
              during Protocol Initialization */
           uint8_t networkFreezerRestore;
} defaultParametersRomOrRam_t;
```

For more information on the APIs, refer to the MiWi Software Design Guide.

3. Migrating MiWi v6.1 to v6.2

This section provides the required information for a customer to migrate MiWi applications implemented on MiWi v6.1 available in the SAM platforms (SAMR21 and SAMR30) to MiWi v6.2.

3.1 New Features and Enhancements of MiWi v6.2

- OTAU support for the SAMR30 device type (supported boards: SAMR30 Xplained Pro and SAMR30 module Xplained Pro)
- Support for OTAU - External memory is added on the SAMR30 module Xplained Pro
- Addition of bootloader files for the SAMR30 boards
- Support for parallel Over-the-Air firmware upgrade
- Mesh enhancement:
  – Addition of acknowledgment wait time for application data in Mesh based on network size
  – Indirect data transmission and keep alive mechanism improvements
- Fix for random RSSI values in application data indication
- Improved Sleep implementation with reduced power consumption
- Duplicate frame rejection for application data handling for Mesh
- Security handling in P2P/Star modified to support memory management module
- Captions updated on the example projects shown in Atmel Studio
- General improvements and bug fixes

3.1.1 OTAU External Memory Support for SAMR30 Module XPRO

The SAMR30 Module Xplained Pro board has AT25DFX041B external Flash. The user has an option to choose this external Flash for storing the Over-the-Air (OTA) received firmware image. This can be done by adding the symbol OTAU_USE_EXTERNAL_MEMORY in the project configuration. For more information, refer to the MiWi Software Design Guide.

3.1.2 Mesh Enhancement - Acknowledgment Wait Time Handling

In MiWi v6.1, FRAME_ACK_WAIT_INTERVAL is used to define the time to wait for the acknowledgment before the next retry of the Mesh frame transmission. This is removed in the release MiWi v6.2, where acknowledgment wait time is calculated based on the time taken to reach one hop, maximum hops based on number of coordinators (using NUM_OF_COORDINATORS), and maximum data request interval of end-devices (using new macro MAXIMUM_DATA_REQUEST_SEND_INTERVAL).

3.1.3 Duplicate Frame Rejection in Mesh

Duplicate frames in Mesh are rejected based on sequence number and source address. When a packet is received at the Mesh network layer, the duplicate rejection table is checked to determine if an entry with the same source address and sequence number already exists. If the entry exists, then the packet will be dropped. If the entry does not exist, then the new entry is created in the duplicate rejection table with the received source address and sequence number. This entry will be removed based on acknowledgment wait time duration. The number of entries kept in the duplicate rejection table can be configured using the macro, DUPLICATE_REJECTION_TABLE_SIZE.

3.2 File Additions

The following are the new files added to the MiWi v6.2 stack:

1. Supporting files for Sleep Manager are available in ASF at `\thirdparty\wireless\miwi\services\sleep_mgr`.
2. Bootloader files for SAMR30 OTAU are available in ASF at `\thirdparty\wireless\miwi\services\otau`.

3.3 MiApp API Changes

The following table lists the API changes with respect to the old APIs.

Table 3-1. API Changes for Old APIs

S. No.	Old APIs (v6.1)	New APIs (v6.2)	Stack Supported
1	`bool MiApp_ProtocolInit(defaultParametersRomOrRam_t *, defaultParametersRamOnly_t *)`	`miwi_status_t MiApp_ProtocolInit(defaultParametersRomOrRam_t *, defaultParametersRamOnly_t *)`	P2P/Star/ Mesh

The return status of `MiApp_ProtocolInit` is changed to handle reconnect case, when a device is not factory new. In Mesh, when a device successfully restores the network parameters, it tries to reconnect with the restored network parameters. This is indicated with the `RECONNECTION_IN_PROGRESS` status in `MiApp_ProtocolInit` API Call. If this status is received, then the application needs to wait for reconnection callback before proceeding to call further MiApp APIs. For more information on the APIs, refer to the MiWi Software Design Guide.

4. Migrating MiWi v6.0 to v6.1

This section describes migration of MiWi applications implemented with MiWi v6.0 available in the SAM platforms (SAMR21 and SAMR30) to MiWi v6.1.

4.1 New Features and Enhancements of MiWi v6.1

1. Network Freezer
2. Improved memory management based on heap
3. Sleep mode
4. Over-the-Air Upgrade (OTAU)
5. Support for SAMR21 modules and SAMR30 module

4.1.1 Network Freezer

The Network Freezer feature saves critical network information into the Nonvolatile Memory (NVM) and restores them after the power cycle. In this way, the application supports the power cycle scenario and the network can be restored to the previous state of the power cycle without many message exchanges after the power cycle.

Additionally, wear-leveling implementation reduces the number of "backup-erase-rewrite" cycles and thereby improves the Flash lifetime.

4.1.1.1 Interface

The Network Freezer feature is enabled by defining `ENABLE_NETWORK_FREEZER` API in the configuration file of the application project. This feature is invoked by calling the MiApp function `MiApp_ProtocolInit`. When Network Freezer is enabled in the application, the network information is restored from NVM; otherwise, the network information in NVM is erased and the wireless node starts as a factory new device.

4.1.1.2 Additional Notes

The Network Freezer feature requires NVM to store the critical network information. In MiWi v6.1 and later, the NVM used for this implementation is the internal Flash.

4.1.2 Sleep Mode

For most of the applications, it is critical to provide long battery life for the sleeping devices. A device can be either in the Active mode or Sleep mode. After being powered-up, a node always starts in the Active mode with its MCU fully turned on.

An application can check whether the stack is allowing sleep or not using the `ENABLE_SLEEP_FEATURE` API. If it allows, then the application can go to sleep at a maximum allowable time by stack for proper operation. In Sleep mode, the RF chip and the MCU are in low-power states and the functionalities required for the MCU to wake up remains active. Thus, the application cannot perform any radio Tx/Rx operations and communicate with external periphery in Sleep mode. The majority of power is consumed when the sleeping device is active, requesting for data and sending data in the duty cycle. Therefore, the device is active on its polling period and this can be controlled using the configuration option. Among all nodes, only end-devices can sleep.

4.1.2.1 Interface

Sleep mode can be enabled by defining `ENABLE_SLEEP_FEATURE` in the configuration file of the application project. For API interface information, refer to the MiWi Software Design Guide.

4.1.3 Over-The-Air Upgrade

For most of the applications, Over-the-Air Upgrade (OTAU) is important to update the node for any fixes after release. The OTAU module is implemented as a manufacturer-specific module which can run parallel with any user application to upgrade the node in the background.

Note: The PC tool, **Atmel WiDBG,** is used for upgrading and provides additional features such as debugging nodes and PHY mode. These features are not supported for MiWi in the v6.1 release.

4.2 File Additions

The following are the new files added to the MiWi v6.1 stack:

1. Supporting files for Network Freezer are available in ASF at `\thirdparty\wireless\miwi\services\pds`.
2. Supporting files for OTAU are available in ASF at `\thirdparty\wireless\miwi\services\otau`.

4.3 MiApp API Changes

The following table lists the API changes with respect to the old APIs.

Table 4-1. API Changes for Old APIs

S. No.	Old APIs (v6.0)	New APIs (v6.1)	Stack Supported
1	Not available	`bool MiApp_SubscribeReConnectionCallback(ReconnectionCallback_t callback)`	P2P/Star/Mesh
2	Not available	`bool MiApp_ResetToFactoryNew(void)`	P2P/Star/Mesh
3	Not available	`bool MiApp_ReadyToSleep(uint32_t* sleepTime)`	Mesh
4	Not available	`bool MiApp_ManuSpecSendData(uint8_t addr_len, uint8_t *addr, uint8_t msglen, uint8_t *msgpointer, uint8_t msghandle, bool ackReq, DataConf_callback_t ConfCallback)`	Mesh
5	Not available	`bool MiApp_SubscribeManuSpecDataIndicationCallback(PacketIndCallback_t callback)`	Mesh
6	Not available	`bool MiApp_IsConnected(void)`	Mesh

For more information on the APIs, refer to the MiWi Software Design Guide.

5. MiWi v6.0

This section describes migration of the MiWi applications implemented on MiWi protocol available in Microchip Libraries for Applications (MLA) to SAM platforms (SAMR21 and SAMR30). The MiWi protocol has been upgraded from MiWi protocol v5.30 to v6.0 with the following features.

1. MiWi protocol is ported to the Advanced Software Framework (ASF) to support easy integration of other components, services and drivers in the application.
2. MiWi P2P/Star is ported to the SAMR21 and SAMR30 platforms.
3. MiWi Mesh is redesigned with new features to support large nodes.
4. MiApp APIs are redesigned to support simple, easy-to-use and reliable data transfer.

5.1 Background of MiWi v6.0

The MiWi is Microchip's proprietary Wireless Networking stack designed to support Low Rate Personal Area Networks (LRPANs).

The MiWi supports the following three network topologies:

- Peer-to-Peer (P2P)
- Star
- Mesh

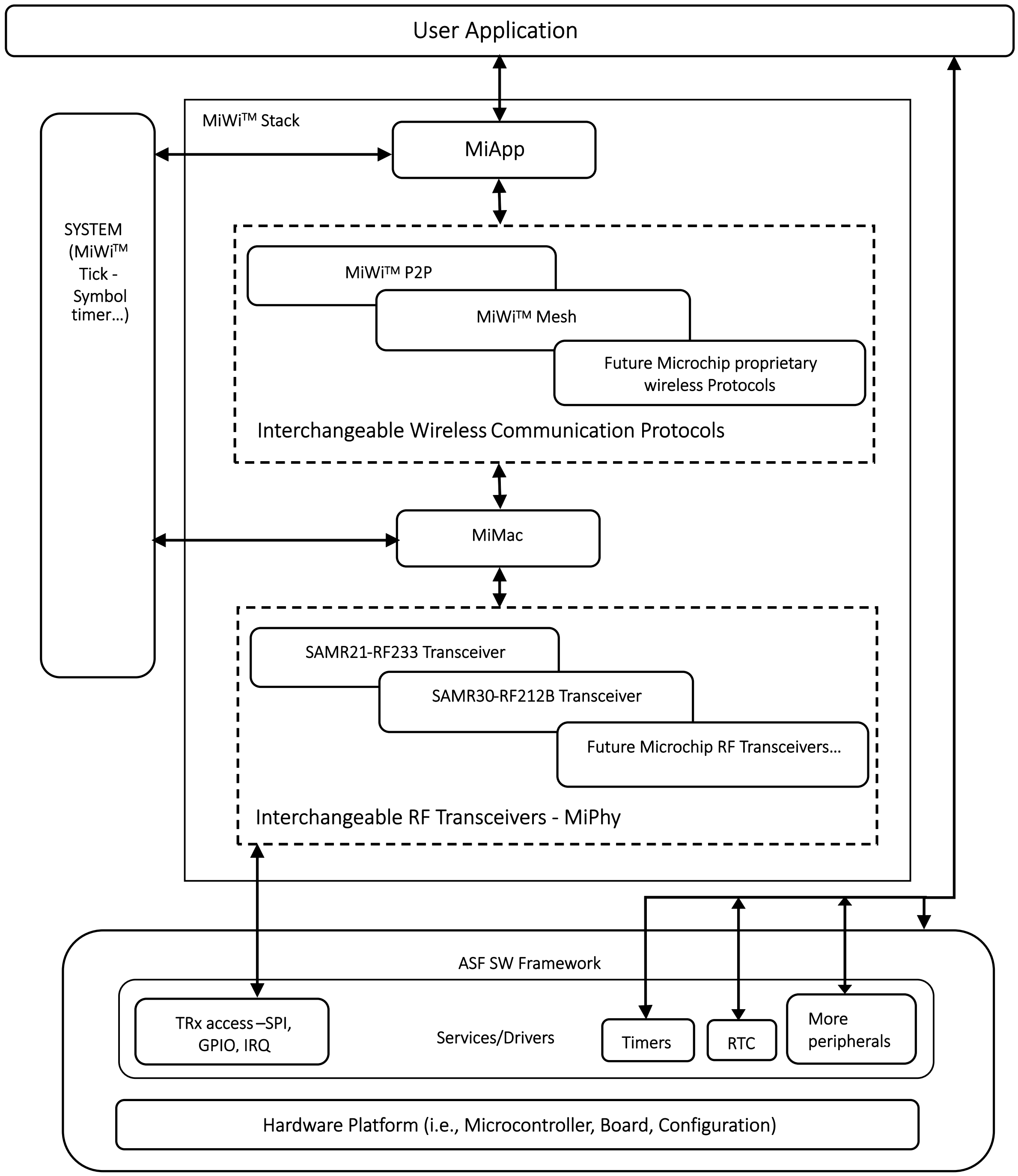
Earlier versions of the MiWi Mesh networking stack (until version 2.10), released in the MiWi protocol v5.30 of Microchip Libraries for Applications (MLA) v2017-03-06, support a library-based Mesh networking stack. However, this is redesigned with the following changes:

1. Optimization of current APIs to improve simplicity.
2. Redesign of the MiWi Mesh with additional features for next generation platforms.
3. A new commissioning procedure to improve the secured inclusion of devices to network.
4. Dynamic switching between device types in the MiWi Mesh.
5. Network secure feature for all network messages.

5.2 Architecture v6.0

The following is the MiWi protocol architecture on ASF, which allows the user to obtain required components, services and drivers from ASF Wizard. For more details, refer to the ASF Wizard of the Atmel Software Framework web page.

Figure 5-1. MiWi™ Architecture



5.3 MiWi Mesh Device Types

The MiWi Mesh protocol supports the following device types:

1. PAN Coordinator
   1.1. Starts the network
   1.2. Assigns and maintains the coordinators and their end-devices' addresses

    1.3. Behaves as coordinator for routing frames
    1.4. Controls the devices which can be included into the network through commissioning
2. Coordinator
    2.1. Joins a network as an end-device
    2.2. Requests PAN coordinator for role upgrade to become a coordinator
    2.3. Supports routing of frames within the network
    2.4. Stores the commissioning information from PAN coordinator and allows only the commissioned devices to participate in the network
    2.5. Maintains its end-devices and their addresses
    2.6. Maintains data for sleeping end-devices
3. End-Device
    3.1. Joins network through available coordinators
    3.2. Supports Rx-On end-device and Sleeping end-device for battery operated devices
    3.3. Supports dynamic switching between Rx-On to sleeping end-deice and vice versa

5.4 MiWi Mesh Frame Format

The network header and application payload of the MiWi Mesh are encapsulated inside the standard IEEE® 805.15.4 data frame payload, but the stack does not adhere to the standard. Therefore, the MiWi Mesh does not receive and process IEEE 805.15.4 command frames. The following figure illustrates a general frame format composed of an IEEE 805.15.4 MAC header, network header, application payload, optional message integrity code (MIC), and a check sum (CRC).

Figure 5-2. General MiWi™ Frame Format

2	1	2	2/8	2	0/2/8	1	1	1	0/2	0/2	0/2/8	0/5	Variable	0/4	2
Frame Control	Sequence number	Dest. PANID	Dest. Address	Source PANID	Source Address	Hops	Frame Control	Sequence number	Dest. PANID	Dest. Address	Source Address	Auxiliary Security Header	Payload	MIC	CRC
MAC Header						Network Header							Payload	Network Footer	

5.4.1 MAC Header – Frame Control

The following figure illustrates the Frame Control field of the MAC header.

Figure 5-3. MAC Header – Frame Control Field

Bits:0-2	3	4	5	6	7:9	10:11	12:13	14:15
Frame Type	Security Enabled	Frame Pending	Ack. Request	PAN ID Compression	Reserved	Dest. Address Mode	Frame Version	Source Address Mode

This is the fixed MAC Frame control field settings used in MiWi Mesh. The following table lists the settings used for a Frame Control field of the MAC header.

Table 5-1. MAC Frame Control Field Settings

Field Name	Settings
Frame Type	Data
Security Enabled	False
Frame Pending	True if pending data available for sleeping end-device or False otherwise
Acknowledgment Request	True for unicast frames and False for broadcast frames

...........continued	
Field Name	**Settings**
PAN ID Compression	True
Destination Addressing Mode	0 for no address fields, 2 for 16-bit short address and 3 for 64-bit extended address
Frame Version	0
Source Addressing Mode	0 for no address fields, 2 for 16-bit short address and 3 for 64-bit extended address

5.4.2 Network Header

5.4.2.1 Hops Field

The Hops field provides the number of hops the packet is allowed to be retransmitted. For example, 00h indicates that the packet is not retransmitted.

5.4.2.2 Frame Control Field

The Frame Control field is a bit map which defines the behavior of a packet, as shown in the following figure.

Figure 5-4. Network Header – Frame Control Field

Bits:0-1	2	3	4	5	6-7
Frame Type	Security Enabled	Infra Cluster	Ack. Request	Address same as MAC	Reserved

The following table details the Frame Control field of the Network Header.

Table 5-2. Network Header Frame Control Field Description

Bit Number	Field Name	Description
6-7	Reserved	Set the bit as '`0`' for this implementation.
5	Address same as MAC	This bit is set when the MAC address fields and Network address fields are the same. This is useful when the sleeping end-device polls the parent for data, with relatively less bytes Over-the-Air for single hop from the network layer.
4	Acknowledgment Request	This bit is set when the source device requests an Network layer acknowledgment of receipt from the destination device.
3	Intra Cluster	Reserved in this implementation. Set the bit as '`1`'.
2	Security Enabled	This bit is set when data packet is encrypted at the application level.
0-1	Frame Type	These bits indicate as follows: • `00` – Data • `01` – Command • `10` – Manufacturer specific • `11` – Reserved

5.4.2.3 Sequence Number field

The Sequence Number field is 1 byte in length and specifies the sequence identifier for the frame. The Sequence Number field shall be increased by 1 for every outgoing frame, originating on the node, and it must not be changed for routed frames.

5.4.2.4 Source Address field

The Source Address field is 2 bytes or 8 bytes in length and specifies the network address of the node originating the frame. Eight bytes are used when security is enabled, where source IEEE address is required for security processing.

5.4.2.5 Destination Address field

The Destination Address field is 2 bytes in length and specifies the network address of the destination node. The Destination Address field can be set for other frames except unicast to a node, as per the following table.

Table 5-3. Network Header Destination Address Field Description

Destination Address Value	Description
0xFFFF	Broadcast to every device
0xFFFE	Multicast to all FFD's
0xFFFD	Multicast to all Coordinators

5.4.2.6 Auxiliary Security Header Field

The Auxiliary Security Header field specifies information required for security processing, including how the frame is protected (security level) and frame counter. This field shall be present only if the Security Enabled sub-field is set to one.

Table 5-4. Auxiliary Security Header Field

Bytes: 1	4	8
Security Level	Frame Counter	Source long address

The supported security levels are as follows:

- 0 (No security)
- 1 (Authentication-4 bytes MIC)
- 4 (Encryption only)
- 5 (Encryption with authentication-4 bytes MIC)

5.5 MiWi Mesh – Device Addressing Mechanism

The MiWi Mesh uses 2 bytes short address to specify nodes in the network when performing routing across the network. The address is allocated during the joining process. The lower byte is used to identify the end-devices. The higher byte is used to identify the coordinators.

Bit 15:8	Bit 7	Bit 6:0
Coordinator identifier	RxOnWhenIdle	End-device identifier

5.6 MiWi Mesh – Networking

The MiWi Mesh network features are categorized as follows:

1. Network commissioning
2. Start and Join network
3. Routing in network

5.6.1 Network Commissioning

The network commissioning controls the devices which can participate in the network.

1. The application on PAN coordinator reads the IEEE address (for example, it can be improved to read from bar code) from one or more devices.
2. PAN coordinator calculates the 64-byte bloom filter value with the read information.
3. Calculated bloom filter value is sent to all the coordinators in the network.
4. Coordinators provide beacon only to the devices which have its IEEE address in the bloom filter.

5.6.2 Start and Join Network

1. Only the PAN coordinator can start a network.
2. Joining device sends beacon request to obtain information about the available networks in its personal operating space.
3. The PAN coordinator or coordinator evaluates the beacon request by parsing the given IEEE address with the bloom filter value. If found, it sends a beacon frame with a beacon payload which includes PAN coordinator hop count and bloom filter value (64 bytes). If not found, it discards the packet.
4. Upon receiving the beacon frames, the joining device parses it and checks its own address in the bloom filter value. It then decides its parent based on associate permit, children capacity and Link Quality Indicator (LQI) of the received beacons. After choosing the parent, it unicasts the Mesh Connection Request packet (includes its capability and JoinWish field) to the selected parent.
   The JoinWish field has two bits, C and ED, and remaining bits are reserved.
   - If both bits are set in JoinWish, then the particular device joins as an end-device if the coordinator capacity is currently unavailable in the network.
   - If only C bit is set, then the device joins as a coordinator only.
   - If only ED bit is set, then the device joins as an end-device only.
5. If the parent is the PAN coordinator and JoinWish field is set with C and ED or C only, then the PAN coordinator checks whether it has a new coordinator address. If available, it sends the Mesh Connection Response with device address as allocated new coordinator address. If the address is unavailable or the JoinWish field has only ED set, then it allocates the end-device address and sends the Mesh Connection Response.
6. If the parent is the coordinator, then it allocates end-device address and sends Mesh Connection Response with the device address as allocated end-device address.
7. The joining device parses the Mesh Connection Response and uses the received network address along with the received network key for further communications in the network.
8. The joining device, which is coordinator capable, receives an end-device address. Based on Role Upgrade Time-out (configurable), the device sends a Role Upgrade Request packet to the PAN coordinator in order to upgrade its role from an end-device to the coordinator.
9. When the PAN coordinator receives a Role Upgrade Request, it checks whether the coordinator address is available. If the address is available, it allocates a new coordinator address and sends the Role Upgrade Response with the allocated address and status as success. If an address is unavailable, then it sends a Role Upgrade Response with a failure status.

5.6.3 Routing in Network

1. During the joining procedure and role upgrade, the route table is updated in all the coordinators.
2. The route table in the coordinators is used to route the packet to the destination device.
3. When the device does not have the next hop address for the destination, it will trigger a broadcast for a route request to the destination.
4. Unlike the legacy route request in AODV routing protocols, the reply is generated from any node which has the next hop information in its routing table.
5. The source device (which initiated the route request) selects the route reply for the destination based on the fewer hops and best LQI.
6. However, to establish and synchronize the network periodically, the route table update is broadcasted to a single hop based on preconfigured intervals.
7. This ensures that the coordinators in the network share the neighbor's information with its neighbors.

5.7 Macros for MiWi Mesh

S.No.	MACRO	Default Value	Range	Description
1	CHANNEL_MAP	(1<<25)	-	Channel map is a bit map used to select appropriate channels for starting or establishing connection in the network. Bit map based on the physical layer. Set/ Clear of any bits in the below range is valid. For 2.4 GHz – 0x07FFF800 For SubGHz – 0x000007FF
2	KEEP_ALIVE_COORDINATOR_SEND_INTERVAL	120	0-255	Time interval in which a coordinator device sends keep alive message to PAN coordinator
3	KEEP_ALIVE_COORDINATOR_TIMEOUT_IN_SEC	KEEP_ALIVE_COORDINATOR_SEND_INTERVAL * 10	0 to (2^32 – 1)	Time-out in seconds of a coordinator at which the PAN coordinator removes the coordinator from the coordinator table
4	KEEP_ALIVE_RXONENDDEVICE_SEND_INTERVAL	120	0-255	Time interval in which an Rx-enabled end-device sends keep alive message to its parent coordinator
5	KEEP_ALIVE_RXONENDDEVICE_TIMEOUT_IN_SEC	KEEP_ALIVE_RXONENDDEVICE_SEND_INTERVAL * 10	0 to (2^32 – 1)	Time-out in seconds of Rx On Enddevice at which the coordinators removes the Rx On Enddevice from the Enddevice table
6	MAX_NUMBER_OF_DEVICES_IN_NETWORK*	32	0 - 200	Table size of the maximum number of devices in the network to generate bloom filter. This is applicable only for PAN coordinator.
7	DEVICE_TIMEOUT1	15000		Time-out of end device reported to coordinator.
8	ROLE_UPGRADE_INTERVAL_IN_SEC	20	0 - 255	Time interval in which a coordinator capable device requests for role upgrade to PAN coordinator
9	CONNECTION_RESPONSE_WAIT_IN_SEC	5	0 - 255	Time interval to wait for connection response from the coordinator
10	NUM_OF_COORDINATORS*	10	0 - 200	Maximum number of coordinators anticipated for which the route table needs to be maintained
11	NUM_OF_NONSLEEPING_ENDDEVICES*	5	0 - 127	Maximum number of Rx-On end-device anticipated for coordinator
12	NUM_OF_SLEEPING_ENDDEVICES*	5	0 - 127	Maximum number of Sleeping end-device anticipated for coordinator.
13	ROUTE_UPDATE_INTERVAL	20	0 - 255	Time interval at which the coordinator sends out route update
14	ROUTE_REQ_WAIT_INTERVAL	5	0 - 255	Time to wait for route replies from other coordinators

...........continued

S.No.	MACRO	Default Value	Range	Description
15	INDIRECT_DATA_WAIT_INTERVAL	10	0 - 255	Time interval to maintain the indirect data for sleeping devices in coordinator.
16	DATA_REQUEST_INTERVAL	5	0 - 255	Time interval to send data request from sleeping end-device to coordinator.
17	FRAME_ACK_WAIT_INTERVAL	5	0 - 255	Time interval to wait for network level acknowledgment.
18	FRAME_RETRY	3	0 - 255	Number of retries to be done for application data frames
19	REBROADCAST_TABLE_SIZE*	10	0 - 255	Size for rebroadcast table to maintain rebroadcasted frames in coordinator
20	REBROADCAST_TIMEOUT	5	0 - 255	Time-out to remove entry from rebroadcast table
21	MAX_BEACON_RESULTS*	5	0 - 255	Maximum number of beacon results to be stored during search connection procedure
22	MESH_SECURITY_LEVEL	5	0 - 7	Security levels for CCM* as defined in IEEE 802.15.4
23	PUBLIC_KEY_DEFAULT	{0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F}	-	Public key of the network used to retrieve network key
24	NETWORK_KEY_DEFAULT	{0x00, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77, 0x88, 0x99, 0xAA, 0xBB, 0xCC, 0xDD, 0xEE, 0xFF}	-	Network key used to transact after successful join to the network

5.8 MiApp API Changes

The following table lists the API changes with respect to old APIs.

Table 5-5. API Changes

S. No.	Old APIs	New APIs	Stack Supported
1	`bool MiApp_ProtocolInit(bool bNetworkFreezer);`	`bool MiApp_ProtocolInit (defaultParametersRomOrRam_t *defaultRomOrRamParams, defaultParametersRamOnly_t *defaultRamOnlyParams)`	P2P/Star/Mesh
2	`bool MiApp_SetChannel(uint8_t Channel)`	`bool MiApp_Set(enum id, uint8_t value)`	P2P/Star/Mesh

...........continued

S. No.	Old APIs	New APIs	Stack Supported
3	`bool MiApp_StartConnection( uint8_t Mode, uint8_t ScanDuration, uint32_t ChannelMap)`	`bool MiApp_StartNetwork( uint8_t Mode, uint8_t ScanDuration, uint32_t ChannelMap, FUNC ConfCallback)`	P2P/Star/Mesh
4	`uint8_t MiApp_SearchConnection(uint8_t ScanDuration, uint32_t ChannelMap)`	`uint8_t MiApp_SearchConnection(uint8_t ScanDuration, uint32_t ChannelMap, FUNC ConfCallback)`	P2P/Star/Mesh
5	`uint8_t MiApp_EstablishConnection(uint8_t ActiveScanIndex, uint8_t Mode)`	`uint8_t MiApp_EstablishConnection(uint8_t Channel, uint8_t addr_len, uint8_t addr, uint8_t Capability_info, FUNC ConfCallback)`	P2P/Star/Mesh
6	`void MiApp_RemoveConnection(uint8_t ConnectionIndex)`	`void MiApp_RemoveConnection(uint8_t ConnectionIndex)`	P2P/Star/Mesh
7	`void MiApp_ConnectionMode(uint8_t Mode)`	`void MiApp_ConnectionMode(uint8_t Mode)`	P2P/Star/Mesh
8	`#define MiApp_FlushTx() {TxData = PAYLOAD_START;}`	`MiApp_SendData(uint8_t addr_len, uint8_t addr, uint8_t len, uint8_t pointer, FUNC ConfCallback)`	P2P/Star/Mesh
9	`#define MiApp_WriteData(a) TxBuffer[TxData++] = a`		P2P/Star/Mesh
10	`bool MiApp_BroadcastPacket(bool SecEn )`		P2P/Star/Mesh
11	`bool MiApp_UnicastConnection(uint8_t ConnectionIndex, bool SecEn)`		P2P/Star/Mesh
12	`bool MiApp_UnicastAddress(uint8_t DestinationAddress, bool PermanentAddr, bool SecEn)`		P2P/Star/Mesh
13	`bool MiApp_MessageAvailable(void)`	`MiApp_SubscribeDataIndicationCallback(FUNC callback)`	P2P/Star/Mesh
14	`void MiApp_DiscardMessage(void)`	Not required	P2P/Star/Mesh
15	`uint8_t MiApp_NoiseDetection(uint32_t ChannelMap, uint8_t ScanDuration, uint8_t DetectionMode, OUTPUT uint8_t NoiseLevel)`	`uint8_t MiApp_NoiseDetection(uint32_t ChannelMap, uint8_t ScanDuration, uint8_t DetectionMode, OUTPUT uint8_t NoiseLevel)`	P2P/Star/Mesh
16	`uint8_t MiApp_TransceiverPowerState(uint8_t Mode)`	`uint8_t MiApp_TransceiverPowerState(uint8_t Mode)`	P2P/Star/Mesh

...........continued

S. No.	Old APIs	New APIs	Stack Supported
17	`bool MiApp_InitChannelHopping( uint32_t ChannelMap)`	`bool MiApp_InitChannelHopping( uint32_t ChannelMap)`	P2P/Star/Mesh
18	`bool MiApp_ResyncConnection(uint8_t ConnectionIndex, uint32_t ChannelMap)`	`bool MiApp_ResyncConnection(uint8_t ConnectionIndex, uint32_t ChannelMap)`	P2P/Star/Mesh
19	`uint8_t Total_Connections(void)`	`uint8_t Total_Connections(void)`	P2P/Star/Mesh
20	`void MiApp_BroadcastConnectionTable()`	`void MiApp_BroadcastConnectionTable()`	Star
21	`bool SW_Ack_SrED(uint8_t )`	`bool SW_Ack_SrED(uint8_t )`	Star
22	`void send_link_status(void)`	`void send_link_status(void)`	Star
23	`void Find_InActiveDevices(void)`	`void Find_InActiveDevices(void)`	Star
24	`void MiApp_leave_network(void)`	`void MiApp_leave_network(void)`	Star
25	`bool MiApp_UnicastStar (bool SecEn)`	`bool MiApp_UnicastStar (bool SecEn)`	Star
26	`void MiApp_SetAddressPan(uint8_t address, uint16_t panid)`	`bool MiApp_Set(enum id, uint8_t value)`	Mesh
27	`bool MiApp_IsMemberOfNetwork(void)`	`bool MiApp_IsMemberOfNetwork(void)`	Mesh
28	`addr_t MiApp_GetParentAddress(void)`	`bool MiApp_Get(enum id, uint8_t value)`	Mesh
29	`void MiApp_RequestData(void)`	Not required	Mesh
30	`bool MiApp_SendDataRequest(void)`	Not required	Mesh
31	`uint16_t MiApp_InitSleepRFDBuffers(uint8_t buffer, uint16_t bufferSize, uint16_t rfdMaxDataSize)`	Not required	Mesh
32	`void MiApp_SetNetworkSecure(bool isSecure)`	`bool MiApp_Set(enum id, uint8_t value)`	Mesh
33	Not available	`bool MiApp_SubscribeDataIndicationCallback(PacketIndCallback_t callback)`	Mesh
34	Not available	`bool MiApp_RoleUpgradeNotification_Subscribe(roleUpgrade_callback_t callback)`	Mesh

...........continued

S. No.	Old APIs	New APIs	Stack Supported
35	Not available	`bool MiApp_Commissioning_AddNewDevice(uint64_t joinerAddress, bool triggerBloomUpdate)`	Mesh
36	Not available	`bool MiApp_SubscribeLinkFailureCallback(LinkFailureCallback_t callback)`	Mesh
37	Not available	`uint16_t MiApp_MeshGetNextHopAddr(uint16_t destAddress)`	Mesh

5.9 Limitations

The list of known limitations is as follows:

1. Functional testing for MiWi Mesh was performed on a network consisting of 32 hops with 1 PAN coordinator and 32 coordinators.
2. The MiWi P2P/Star protocol frame format and examples are compatible with earlier versions of MiWi P2P/Star.
3. The 6.0 and later versions of the MiWi Mesh protocol is not interoperable or backward compatible to the earlier versions of MiWi Mesh protocol.

6. Document Revision History

Revision	Date	Section	Description
C	11/2019	1. Migrating MiWi v6.3 to v6.4	Added new section
B	08/2019	2. Migrating MiWi v6.2 to v6.3	Added new section
A	03/2019	Document	Initial revision

The Microchip Website

Microchip provides online support via our website at http://www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to http://www.microchip.com/pcn and follow the registration instructions.

Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: http://www.microchip.com/support

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Legal Notice

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Trademarks

The Microchip name and logo, the Microchip logo, Adaptec, AnyRate, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, chipKIT, chipKIT logo, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PackeTime, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TempTrackr, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, FlashTec, Hyper Speed Control, HyperLight Load, IntelliMOS, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet-Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, Vite, WinPath, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BlueSky, BodyCom, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, INICnet, Inter-Chip Connectivity, JitterBlocker, KleerNet, KleerNet logo, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, SAM-ICE, Serial Quad I/O, SMART-I.S., SQI, SuperSwitcher, SuperSwitcher II, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

The Adaptec logo, Frequency on Demand, Silicon Storage Technology, and Symmcom are registered trademarks of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

ISBN: 978-1-5224-5284-3

Quality Management System

For information regarding Microchip's Quality Management Systems, please visit http://www.microchip.com/quality.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
Corporate Office 2355 West Chandler Blvd. Chandler, AZ 85224-6199 Tel: 480-792-7200 Fax: 480-792-7277 Technical Support: http://www.microchip.com/support Web Address: http://www.microchip.com **Atlanta** Duluth, GA Tel: 678-957-9614 Fax: 678-957-1455 **Austin, TX** Tel: 512-257-3370 **Boston** Westborough, MA Tel: 774-760-0087 Fax: 774-760-0088 **Chicago** Itasca, IL Tel: 630-285-0071 Fax: 630-285-0075 **Dallas** Addison, TX Tel: 972-818-7423 Fax: 972-818-2924 **Detroit** Novi, MI Tel: 248-848-4000 **Houston, TX** Tel: 281-894-5983 **Indianapolis** Noblesville, IN Tel: 317-773-8323 Fax: 317-773-5453 Tel: 317-536-2380 **Los Angeles** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **Raleigh, NC** Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000 **San Jose, CA** Tel: 408-735-9110 Tel: 408-436-4270 **Canada - Toronto** Tel: 905-695-1980 Fax: 905-695-2078	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870 **Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4450-2828 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-72400 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560 **Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-72884388 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654 **UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820