MercuryAPI 1.31.1 Release Notes

Includes release notes for Universal Reader Assistant v4.1.36 and the Autonomous Configuration Tool, v1.3.1. This API release is fully qualified for the entire ThingMagic product line.

These release notes describe the features of MercuryAPI SDK version 1.31.1 relative to version 1.31.0. The release notes for previous versions, going back to 1.29.4 (the last time the MercuryAPI Programmers Guide was updated) are included.

The MercuryAPI SDK includes the .NET (C#) API, the Java API, the C-API, the Universal Reader Assistant demo program written in C#, and the Autonomous Configuration Tool written in Java. For a full description of how to use the API and sample applications, refer to the following documentation:

- The Mercury API Programmers Guide
- Universal Reader Assistant User Guide
- Autonomous Configuration Tool User Guide
- Sargas On-Reader Programming Application Note

All are available at https://www.jadaktech.com/documentation

Note that although Universal Reader Assistant and the Autonomous Configuration Tool are distributed as part of the API, their features and changes are in a separate section in this document.
Autonomous Configuration Tool version 1.2.1 ................................................................. 27
Outstanding Issues and Caveats ....................................................................................... 28
MercuryAPI v1.29.2 ........................................................................................................ 28
Universal Reader Assistant 3.2.1 ................................................................................... 30
Autonomous Configuration Tool version 1.2.1 ............................................................... 30
Compatible Firmware

MercuryAPI version 1.31.1 has been designed to support all the features of the following firmware releases.

- M6e module firmware version 1.21.2
- Micro/Micro-LTE modules, and USBPro reader, firmware version 1.B.2
- Nano module firmware version 1.9.0
- Sargas reader firmware version 5.3.0
- M6 and Astra-EX reader firmware version 4.19.3
- M5e/M5e-Compact, Vega and USB Plus+ readers, firmware version 1.7.5

All features are not available for all products due to firmware or hardware limitations. These release notes will note which products support the new features. Information about support for existing features are found in the MercuryAPI Programmers Guide.
Features Matrix

The following table explains on the list of the features implemented during this release and supported products for each feature.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Supported Modules/Readers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support for Initial Q Setting</td>
<td>Micro, Nano</td>
</tr>
<tr>
<td>Option to send Select with every Query</td>
<td>Micro, Nano</td>
</tr>
<tr>
<td>Support for multiple selects</td>
<td>Micro, Nano</td>
</tr>
<tr>
<td>Support for NXP’s UCODE8 Brand Identifier</td>
<td>Micro</td>
</tr>
<tr>
<td>Generate GEN2V2 iChallenge in FW</td>
<td>Micro, Nano</td>
</tr>
<tr>
<td>Embedded TagOp Support for Read-After-Write Option</td>
<td>Micro</td>
</tr>
<tr>
<td>Japan region changes</td>
<td>Micro, Nano</td>
</tr>
<tr>
<td>Support for configurable CW on/off time/Regulatory Test Revamp</td>
<td>Micro, Nano</td>
</tr>
<tr>
<td>Support for Per-Antenna On-Time Settings</td>
<td>Micro, Nano</td>
</tr>
<tr>
<td>Support for 5 new regions</td>
<td>Nano</td>
</tr>
<tr>
<td>Antenna Multiplexing and GPIOs Implementation in URA</td>
<td>M6e, Micro, Nano</td>
</tr>
<tr>
<td>URA UI Changes</td>
<td>(URA-specific)</td>
</tr>
</tbody>
</table>
New Features of API

The new features of the API are described briefly below. API version v1.31.1 is now the recommended release for all ThingMagic readers users.

See the *MercuryAPI Programmer’s Guide* for more information:

**Support for Initial Q Setting**

One of the assumptions that the writers of the RAIN gen2 protocol made is that if multiple tags responded in the same inventory response slot, each one was extremely unlikely to pick the same number from 0 to 65535 so that when the reader acknowledges one tag’s response with that number, the “losing” tags would realize their failure and try again the very next inventory round.

We have discovered that, in practice, this algorithm is not fool-proof. By default, our module always offers the same number of inventory slots as the previous inventory round and adjusts them in subsequent inventory rounds as more or fewer tags respond. However, when conditions cause hundreds of tags to suddenly respond, many tags respond in the same few slots and the probability that no two tags will pick the same random number is surprisingly low. (There is a 75% chance that two of 430 tags will pick the same random number between 0 and 65535 if only one slot is available). After a long period of no tags responding, we offer as few as 4 slots, which reduces this probability by 25% (to 0.19%) of two tags selecting the same random number. This means that it is highly likely that a tag will erroneously believe that it responded correctly to an inventory round and then not respond again until its session timer expires (which can be several minutes under Session 2 or Session 3 rules). For some applications, this cannot be tolerated. Min and Max values of “Initial Q” are 2 to 10.

We have made the following changes to reduce, but not entirely eliminate, this issue.

1. Extended the range of dynamic “Q” values from 2 (4 slots) to 10 (1024 slots) under normal operation.
2. Added an option to set “Initial Q” whenever we believe that it is likely a large number of tags will respond.

“Initial Q” would be imposed under the following circumstances:

1. Use whenever there has been a reboot; so, no previous “Q” is available (default is “2” if no “Initial Q” is set).
2. When in the middle of a read cycle, would be used instead of the previous “Q” whenever the target has changed from B to A or A to B.
3. Use at the start of a continuous read cycle or timed read.

“Initial Q” would not be imposed in the following cases:

1. At each asynchronous cycle when continuously reading.
2. Whenever there is a change of antenna. (It is likely all antennas are pointed at the same tag population, so a sudden increase in tag responses is not expected.)

The new parameter in the API to set the “Initial Q” functionality is:

**TMR_PARAM_GEN2_INITIAL_Q** (or) **"/reader/gen2/initQ"**

**C API:**

```c
TMR_GEN2_initQ initQ;
initQ.qEnable = 1;
initQ.qValue = 8;

ret = TMR_paramSet(rp, TMR_PARAM_GEN2_INITIAL_Q, &initQ);
checkerr(rp, ret, 1, "setting initialQ");

ret = TMR_paramGet(rp, TMR_PARAM_GEN2_INITIAL_Q, &initQ);
checkerr(rp, ret, 1, "Getting initialQ");
Printf("initialQ = %d \n", qValue);
```

**Java API:**

```java
// To get the default value of InitialQ
Gen2.InitQ q1 = (Gen2.InitQ)r.paramGet(TMConstants.TMR_PARAM_GEN2_INITIAL_Q);
System.out.println(q1.toString());

// To configure the initialQ value, user must set this param.
Gen2.InitQ q = new Gen2.InitQ();
q.qEnable = true;
q.initialQ = 5;
r.paramSet(TMConstants.TMR_PARAM_GEN2_INITIAL_Q, q);
```

**C# API:**

```csharp
// To get the default value of InitialQ
Gen2.InitQ q = (Gen2.InitQ)r.ParamGet("/reader/gen2/initQ");
Console.WriteLine(q);

// To configure the initialQ value, user must set this param.
Gen2.InitQ p = new Gen2.InitQ();
p.qEnable = true;
p.initialQ = 6;
r.ParamSet("/reader/gen2/initQ",p);
```

**Option to send Select with every Query**

By default, we send a Select command to a population of tags when reading starts and each time an antenna is switched. This is to minimize the overhead of sending this relatively long message. Once a tag is selected for an inventory round, it is designed to remember this until it is
successfully inventoried during that or any successive inventory round. This works under normal circumstances, but not when tags frequently enter and leave the read field. A tag that is not present to hear the Select command will respond during the first opportunity, not knowing that it is not “qualified” to do so based on the Select criteria.

Also, tags at the very limit of their receive sensitivity may not hear the Select, but correctly interpret a Query of a subsequent inventory round and respond inappropriately.

To accommodate applications requiring more frequent Selects, we have added a setting to send the Select command with every Query (which begins an inventory round).

In the API, the parameter to enable sending Select with every Query (start of inventory round) is:

**TMR_PARAM_GEN2_SEND_SELECT (or) "/reader/gen2/sendSelect"**

By default, the value of this flag is false(0x00) which means Select will not be sent with every Query. A value of true(0x01) will enable Select-with-every-Query.

Note that sending Select with every Query is the default behavior when Fast Search mode is enabled, but this mode also makes other changes to the search algorithm, which may not be desirable for all applications.

**C API:**

```c
bool select;
select = 1;
ret = TMR_paramSet(rp, TMR_PARAM_GEN2_SEND_SELECT, &select);
checkerr(rp, ret, 1, "setting select with query");
ret = TMR_paramGet(rp, TMR_PARAM_GEN2_SEND_SELECT, &select);
checkerr(rp, ret, 1, "Getting select with query");
printf("\n select value = %d\n", select);
```

**Java API:**

```java
// To send select with every query
r.paramSet(TMConstants.TMR_PARAM_GEN2_SEND_SELECT, true);

// To disable, sending select with every query command
r.paramSet(TMConstants.TMR_PARAM_GEN2_SEND_SELECT, false);

// To get the status of the param value
System.out.println("value: "+ r.paramGet(TMConstants.TMR_PARAM_GEN2_SEND_SELECT);
```

**C# API:**

```csharp
// To send select with every query
```
r.ParamSet("/reader/gen2/sendSelect", true);

// To disable, sending select with every query command
r.ParamSet("/reader/gen2/sendSelect", false);

// To get the status of the param value
System.out.println("value: "+ r.paramGet("/reader/gen2/sendSelect"));

Support for Complex Selects

Traditionally, Select has been used for one purpose: to select among a population of tags or select all but a population of tags based on a desired (or undesired) value in memory. As tags gain more custom features, tag IC designers have found it convenient to use unusual, but legal forms of Select to alert all their tags to perform a proprietary function. To accommodate these features, we now support the full range of Select options that are mandated by the RAIN Gen2 specification.

The two fields that represent this additional functionality are Target (which flag to change) and Action (how to set, reset, flip, or leave alone flags) when the tag memory value either matches the criteria or do not match the criteria.

Here is the list of Action values with respect to matching and non-matching tags: -
<table>
<thead>
<tr>
<th>Action</th>
<th>Matching Tags</th>
<th>Non-Matching Tags</th>
<th>API Enumeration</th>
<th>Behavior if Flag is “SL” (Selected)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Assert Target</td>
<td>De-Assert Target</td>
<td>ON_N_OFF</td>
<td>Matching tags will respond, and Non-Matching tags will NOT respond</td>
</tr>
<tr>
<td>1</td>
<td>Assert Target</td>
<td>Do Nothing</td>
<td>ON_N_NOP</td>
<td>Matching tags will respond, and Non-Matching tags will respond based on previous SL flag status from last Action</td>
</tr>
<tr>
<td>2</td>
<td>Do Nothing</td>
<td>De-Assert Target</td>
<td>NOP_N_OFF</td>
<td>Matching tags will respond based on previous SL flag status from last Action and Non-Matching tags will NOT respond</td>
</tr>
<tr>
<td>3</td>
<td>Negate Target</td>
<td>Do Nothing</td>
<td>NEG_N_NOP</td>
<td>Previous SL flag will be nullified for Matching tags and Non-Matching tags will respond based on previous SL flag status from last Action</td>
</tr>
<tr>
<td>4</td>
<td>De-Assert Target</td>
<td>Assert Target</td>
<td>OFF_N_ON</td>
<td>Matching tags will NOT respond and Non-Matching tags will respond based on previous SL flag status from last Action</td>
</tr>
<tr>
<td>5</td>
<td>De-Assert SL</td>
<td>Do Nothing</td>
<td>OFF_N_NOP</td>
<td>Matching tags will NOT respond and Non-Matching tags will respond based on previous SL flag status from last Action</td>
</tr>
<tr>
<td>6</td>
<td>Do Nothing</td>
<td>Assert Target</td>
<td>NOP_N_ON</td>
<td>Matching tags will respond based on previous SL flag status from last Action and Non-Matching tags will respond</td>
</tr>
<tr>
<td>7</td>
<td>Do Nothing</td>
<td>Negate Target</td>
<td>NOP_N_NEG</td>
<td>Matching tags will respond based on previous SL flag status from last Action and previous SL flag will be nullified for Non-Matching tags</td>
</tr>
</tbody>
</table>

In the API, the Gen2.Select constructor has not changed. For example, in Java:

```java
Gen2.Select(boolean invert, Gen2.Bank bank, int bitPointer, int bitLength, byte[] mask)
```

What has changed in the API is the optional inclusion of target and action settings:

- Gen2.Select.Target
- Gen2.Select.Action

Actions are specified using the shorthand enumerations in the table above.

Target flags have the following enumerations:

- Inventoried_S0
- Inventoried_S1
- Inventoried_S2
• Inventoried_S3
• Select (SL Flag – the default)

For backward compatibility, we need to support a parameter we introduced to the Select definition, called “Invert”, which allowed a Select criteria to define which tags should not respond rather than those that would. Each one of the Actions has an opposite action as shown in this table:

<table>
<thead>
<tr>
<th>Action set by user</th>
<th>Action set in FW if Invert flag is enabled</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
</tr>
</tbody>
</table>

If the optional Target and Action parameters are not set, the behavior is as before:
• When “Invert” flag is not set, default value of Target = 4 and Action = 0.
• When “Invert” flag is set, default value of Target = 4 and Action = 4.

Refer API methods in “Support for Multiple Selects” feature for optional inclusion of target and action settings in all APIs.

**Support for Multiple Selects**

RAIN tags support the ability to process multiple Select commands to combine individual populations of tags (based on common data fields) into a single group for purposes of responding to subsequent inventory operations. Until recently, there was little need for this feature, but emerging sensor tags now use an initial Select to enable sensor reading as well as an additional Select (or more) to restrict which sensor tags respond.

Sensor tag chips may have temperature sensor, humidity sensor, gas sensor, resistance sensor, capacitance sensor, etc. They are differentiated by address pointers. So, the first Select is used to identify and “wake up” the sensor. If there is only one tag in field for the reader to poll, one Select is all that is needed. However, if it is necessary to select a tag or tags from a group of tags to read sensor data from, an additional Select (or Selects) are needed.

The maximum number of selects that will be allowed are 3. This number of Selects would be required to simultaneously:
1. Determine via the TID field that the tag is of the desired sensor tag make and model
2. Enable tag sensing
3. Isolate only a portion of the sensor tags that are in the read field at that time
A new option has been added in module FW to enable multiple selects.

`GEN2_SELECT_OPTION_MULTIPLE_SELECTS = (1 << 10)`

Multiple selects are expressed as a MultiFilter (a new subclass of TagFilter). These can be used in all the same places as an existing TagFilter, so it is a simple addition to the user-visible API.

Refer to “filter” sample codelet in the MercuryAPI SDK to test multi filter functionality.

C API:

```c
TMR_TagFilter tidFilter, epcFilter, userFilter, *filterArray[3];
TMR_MultiFilter filterList;
uint8_t mask1[2], mask2[2], mask3[2];

filterList.tagFilterList = filterArray;
filterList.len = 0;
{
    /* This select filter matches all Gen2 tags where bits 00-16 of the USER memory are 0x0123 */
    mask1[0] = 0x01;
    mask1[0] = 0x23;
    TMR_TF_init_gen2_select(&userFilter, false, TMR_GEN2_BANK_USER, 0, 16, mask1);
    epcFilter.u.gen2Select.target = SELECT;
    epcFilter.u.gen2Select.action = 0x00;
}
{
    /* This select filter matches all Gen2 tags where bits 08-12 of the TID memory are 0x0030 (that is, tags Manufactured by Alien Technology). */
    mask2[0] = 0x00;
    mask2[1] = 0x30;
    TMR_TF_init_gen2_select(&tidFilter, false, TMR_GEN2_BANK_TID, 8, 12, mask2);
    tidFilter.u.gen2Select.target = SELECT;
    tidFilter.u.gen2Select.action = ON_N_OFF;
}
{
    /* This select filter matches all Gen2 tags whose EPC starts with 0xABAB */
    mask3[0] = 0xAB;
    mask3[1] = 0xAB;
    TMR_TF_init_gen2_select(&epcFilter, false, TMR_GEN2_BANK_EPC, 32, 16, mask3);
    epcFilter.u.gen2Select.target = SELECT;
    epcFilter.u.gen2Select.action = NOP_N_OFF;
}

/* Assemble two filters in filterArray */
filterArray[filterList.len++] = &userFilter;
filterArray[filterList.len++] = &tidFilter;
```
filterArray[filterList.len++] = &epcFilter;

/* Assign TMR_GEN2_MultiSelect filter to TMR_TagFilter filter */
filter.type = TMR_FILTER_TYPE_MULTI;
filter.u.multiFilterList = filterList;
TMR_RP_set_filter(&Plan, &filter);

Java API:

// create and initialize tid filter
    Gen2.Select tidFilter = new Gen2.Select(false, Gen2.Bank.TID, 32, 16, new byte[]
    {(byte)0x01,(byte)0x2F});
    tidFilter.target = Gen2.Select.Target.Select;
    tidFilter.action = Gen2.Select.Action.ON_N_OFF;

// create and initialize epc filter
    Gen2.Select epcFilter = new Gen2.Select(false, Gen2.Bank.EPC, 32, 16, new byte[]
    {(byte)0xAA,(byte)0xAA});
    epcFilter.target = Gen2.Select.Target.Select;
    epcFilter.action = Gen2.Select.Action.ON_N_OFF;

// create and initialize user filter
    Gen2.Select userFilter = new Gen2.Select(false, Gen2.Bank.USER, 32, 16, new byte[]
    {(byte)0x00,(byte)0x00});
    userFilter.target = Gen2.Select.Target.Select;
    userFilter.action = Gen2.Select.Action.ON_N_OFF;

// Initialize multiFilter by passing tidFilter, epcFilter and userFilter as elements to TagFilter array
    MultiFilter multiFilter = new MultiFilter(new TagFilter[]{tidFilter, epcFilter, userFilter});

// Pass this multiFilter in simple readplan
    SimpleReadPlan plan = new SimpleReadPlan(antennaList, TagProtocol.GEN2, multiFilter, null, 1000);

C# API:

Gen2.Select tidFilter = new Gen2.Select(false, Gen2.Bank.TID, 32, 16, new byte[]
{(byte)0x01, (byte)0x3F});
    tidFilter.target = Gen2.Select.Target.Select;
    tidFilter.action = Gen2.Select.Action.ON_N_OFF;

Gen2.Select epcFilter = new Gen2.Select(false, Gen2.Bank.EPC, 32, 16, new byte[]
{(byte)0xAB, (byte)0xCD});
    epcFilter.target = Gen2.Select.Target.Select;
    epcFilter.action = Gen2.Select.Action.ON_N_OFF;
Gen2.Select userFilter = new Gen2.Select(false, Gen2.Bank.USER, 32, 16, new byte[]{(byte)0x64, (byte)0x75});
epcFilter.target = Gen2.Select.Target.Select;
epcFilter.action = Gen2.Select.Action.ON_N_OFF;

//Initialize multifier with filterarray
MultiFilter multiFilter = new MultiFilter(new TagFilter[]{tidFilter, epcFilter, userFilter});

SimpleReadPlan plan = new SimpleReadPlan(antennaList, TagProtocol.GEN2, multiFilter, rData, 1000);
r.ParamSet("/reader/read/plan", plan);

**Support for NXP’s UCODE8 Brand Identifier**

This feature will allow brand owners to implement a product originality check for products tagged with NXP UCODE8-based Tags. Customers of NXP are granted a dedicated unique 16-bit brand identifier which is programmed during the manufacturing process by NXP and is unalterable in the field.

A Select command on address bit 204h (with mask length = 1 and mask = 0x80) in the EPC memory bank will trigger the UCODE8/8m tag to respond in the inventory round with PC + EPC + Brand Identifier + CRC as if it were one long EPC. To prevent cloning, the Brand Identifier is scrambled. The module FW automatically unscrambles the Brand Identifier. The resultant decoded Brand Identifier will be sent as part of tag metadata if Brand Identifier metadata flag is enabled. At the host, the Brand identifier can be received and the originality check completed.

Default value of the UCODE8/8m tag’s Brand Identifier is "AAAAh". Customer specific Brand Identifiers can be requested from NXP by ordering a dedicated product code.

If Select is not used, then this field will be skipped and no Brand Identifier information will be returned even if the Brand Identifier metadata flag is enabled.

In the API, new metadata flag named "BRAND_IDENTIFIER(0x1000)" has been added in TagMetadata enumeration to support the functionality of Brand identifier feature. Refer "read" sample codelet in the MercuryAPI SDK to test this functionality.

**Generate GEN2V2 iChallenge in FW**

Initially, it was the host’s responsibility to input an iChallenge value to perform GEN2V2 tag authentication operations. But in latest firmware versions, iChallenge will be generated in module FW randomly without having the need for the host to provide it as an input. So, corresponding changes have been done in API. The change to the API is that iChallenge is no longer needed for both TAM1 and TAM2 authentication methods.
C API Changes:

**TAM1 Method:**
TMR_Status
TMR_TagOp_init_GEN2_NXP_AES_Tam1authentication(TMR_TagOp_GEN2_NXP_Tam1Authentication *auth,
TMR_NXP_KeyId keyid, TMR_uint8List *key, bool sendRawData )

**TAM2 Method:**
TMR_Status
TMR_TagOp_init_GEN2_NXP_AES_Tam2authentication(TMR_TagOp_GEN2_NXP_Tam2Authentication *auth,
TMR_NXP_KeyId keyid, TMR_uint8List *key, TMR_NXP_Profile profile, uint16_t Offset,uint8_t blockCount, int protMode, bool sendRawData)

Java API Changes:

**TAM1 Method:**
public Tam1Authentication(AES.KeyId keyId ,byte[] key, boolean enableRawData)

**TAM2 Method:**
public Tam2Authentication(AES.KeyId keyId ,byte[] key, AES.Profile profile,
int offset, int blockCount, int protMode, boolean enableRawData)

C# API Changes:

**TAM1 Method:**
public Tam1Authentication(KeyId keyid, ushort[] key, bool sendRawData)

**TAM2 Method:**
public Tam2Authentication(KeyId keyid, ushort[] key, Profile profile,
ushort offset, ushort blockCount, ushort protMode, bool sendRawData)

: base(keyid, key, sendRawData)

Note: - These changes were not implemented for Sargas/Izar firmware. So, we insist the users to use previously released API version v1.29.4 to work with Gen2v2 Authenticate tag operations on Sargas and Izar. (Ref #5927).

**Backward Compatibility Issue:**
Gen2v2 tag operations will not work with cross release builds. i.e. with previous released FW version & current released API version and vice-versa, as various changes have been done in the APIs corresponding to FW modifications. (Ref #5927).

<table>
<thead>
<tr>
<th>API Version</th>
<th>Firmware Version</th>
<th>Gen2V2 Tag Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>API current released version v1.31.1</td>
<td>Micro current released version v1.B.1</td>
<td>Will work</td>
</tr>
<tr>
<td>API current released version v1.31.1</td>
<td>Nano current released version v1.9.0</td>
<td>Will work</td>
</tr>
</tbody>
</table>
### Embedded TagOp Support for Read-After-Write Option

Embedded tag operation support has been added in this API release for read-after-write option in both EPC write and Data write operations. Earlier API releases only supported this function for standalone tag operations. (Both functions are performed without dropping the RF signal, as is required for some sensor tags which use the write operation to trigger sensor value updating, which is subsequently read.)

User can achieve this functionality through two operations:

1. **Gen2.WriteData + Gen2.ReadData**
2. **Gen2.WriteTag + Gen2.ReadData**

For executing embedded operation:

1. Create a TagOpList.
2. Add either of the above-mentioned operations to the list.
3. Set a simple read plan by passing the tagoplist as the tagop.
4. Execute `r.read(5000)` to execute the embedded operation.

Refer to “WriteTag” sample codelet in the MercuryAPI SDK to test this functionality.

### Japan Region Changes

- A new JP3 region support has been added for Micro and Nano modules with 6 channels and power limited to +24 dBm. So, corresponding changes have been done in API, new enum named JP3 with value 0x18 has been added to “TMR_Region_enum”.

- The number of channels in the JP2 region channels has been reduced from 19 to 13. (The highest 6 channels were removed due to their recent allocation to a different class of RFID service, according to Japanese regulations). No API changes are required.

<table>
<thead>
<tr>
<th>API current released version</th>
<th>Micro previous released version</th>
<th>Does’t work</th>
</tr>
</thead>
<tbody>
<tr>
<td>v1.31.1</td>
<td>v1.9.1</td>
<td></td>
</tr>
<tr>
<td>API current released version</td>
<td>Nano previous released version</td>
<td>Doesn’t work</td>
</tr>
<tr>
<td>v1.31.1</td>
<td>v1.7.3</td>
<td></td>
</tr>
<tr>
<td>API previous released version</td>
<td>Micro current released version</td>
<td>Doesn’t work</td>
</tr>
<tr>
<td>v1.29.4</td>
<td>v1.8.1</td>
<td></td>
</tr>
<tr>
<td>API previous released version</td>
<td>Micro current released version</td>
<td>Doesn’t work</td>
</tr>
<tr>
<td>v1.29.4</td>
<td>v1.9.0</td>
<td></td>
</tr>
<tr>
<td>API previous released version</td>
<td>Micro previous released version</td>
<td>Will work</td>
</tr>
<tr>
<td>v1.29.4</td>
<td>v1.9.1</td>
<td></td>
</tr>
<tr>
<td>API previous released version</td>
<td>Nano previous released version</td>
<td>Will work</td>
</tr>
<tr>
<td>v1.29.4</td>
<td>v1.7.3</td>
<td></td>
</tr>
</tbody>
</table>
- The Max power for JP region is now limited to +30 dBm (instead of +31.5 dBm) per Japanese regulations. No API changes are required.

The comparison table summarizes all details:

<table>
<thead>
<tr>
<th>Region Name</th>
<th>JP</th>
<th>JP2</th>
<th>JP3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Country or Region Covered</td>
<td>Japan</td>
<td>Japan</td>
<td>Japan</td>
</tr>
<tr>
<td>Modules Supported</td>
<td>Micro</td>
<td>Micro</td>
<td>Micro</td>
</tr>
<tr>
<td></td>
<td>Nano</td>
<td>Nano</td>
<td>Nano</td>
</tr>
<tr>
<td>Region Code</td>
<td>0x05</td>
<td>0x17</td>
<td>0x18</td>
</tr>
<tr>
<td>Lowest Freq Permitted in Band</td>
<td>916800</td>
<td>916800</td>
<td>916800</td>
</tr>
<tr>
<td>Highest Freq Permitted in Band</td>
<td>920800</td>
<td>923400</td>
<td>923400</td>
</tr>
<tr>
<td>Smallest Step Size between</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Channels</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum Dwell Time On any</td>
<td>4 sec</td>
<td>4 sec</td>
<td>4 sec</td>
</tr>
<tr>
<td>Channel</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max RF Power</td>
<td>+30 dBm</td>
<td>+27 dBm</td>
<td>+24 dBm</td>
</tr>
<tr>
<td></td>
<td>(Micro)</td>
<td>(Nano)</td>
<td></td>
</tr>
<tr>
<td>Carrier Sense Level</td>
<td>-74 dBm</td>
<td>-74 dBm</td>
<td>-74 dBm</td>
</tr>
<tr>
<td>Lowest Chan in Hop Table</td>
<td>916800</td>
<td>916800</td>
<td>916800</td>
</tr>
<tr>
<td>Highest Chan in Hop Table</td>
<td>920800</td>
<td>922200</td>
<td>920800</td>
</tr>
<tr>
<td></td>
<td>921400</td>
<td>922200</td>
<td>921400</td>
</tr>
<tr>
<td></td>
<td>922000</td>
<td>922000</td>
<td>922000</td>
</tr>
<tr>
<td></td>
<td>921200</td>
<td>921200</td>
<td>921200</td>
</tr>
<tr>
<td></td>
<td>921800</td>
<td>921800</td>
<td>921800</td>
</tr>
<tr>
<td></td>
<td>921000</td>
<td>921000</td>
<td>921000</td>
</tr>
<tr>
<td></td>
<td>921600</td>
<td>921600</td>
<td>921600</td>
</tr>
<tr>
<td>Hop Table</td>
<td>920800</td>
<td>920800</td>
<td>920800</td>
</tr>
<tr>
<td></td>
<td>919200</td>
<td>920600</td>
<td>919200</td>
</tr>
<tr>
<td></td>
<td>920600</td>
<td>920400</td>
<td>920600</td>
</tr>
<tr>
<td></td>
<td>916800</td>
<td>916800</td>
<td>916800</td>
</tr>
<tr>
<td></td>
<td>918000</td>
<td>919200</td>
<td>918000</td>
</tr>
<tr>
<td></td>
<td>920400</td>
<td>918000</td>
<td>920400</td>
</tr>
</tbody>
</table>

Support for Configurable CW ON/OFF Time (Regulatory Test Features)

Previously, at the module FW level, a CW (continuous wave) signal could be turned on and off, with no time limit. A PRBS (randomly modulated) signal could be turned on, but only for a configured time limit. (Universal Reader Assistant artificially imposed a time limit setting on CW, which is not in the firmware or API.)
Now support has been added for much more flexibility for both CW and PRBS signal generation. Either can be enabled or disabled for an arbitrary amount of time and, while enabled, they can operate on an on-off duty cycle.

This scope capture of signal amplitude shows CW signal with 256ms RF ON time and 96ms RF OFF time.

This scope capture of signal amplitude shows PRBS signal with 256ms RF ON time and 240ms RF OFF time.
The module needs 2ms RF settling time after ramp up. So, if RF ON time is specified as 2ms or less, RF will not be turned on. 3ms is the least time where RF ON pulse can be seen. For RF ON time > 2ms, a compensation of 2ms for settling time is included in FW for accurate timing.

RF off time is also compensated 1ms for power supply settling. So, RF is turned off only if RF off time > 1ms.

If RF on time is > 2ms and off time is 1ms or less, then RF will remain on continuously for the specified number of on/off cycles times on time.
Valid min and max limits of RF ON and OFF time are 2 to 65535 (0xFFFF) milliseconds. (0 and 1 are permitted, but not recommended.)

The FW will monitor module temperature when CW/PRBS is running continuously to avoid module heating up. If the limit of internal temperature is exceeded, a 0x504 error code will be returned in response to the CmdGetTemperature Method. (Periodic temperature status messages are not send when the module is in this mode, so it must be polled.)

The channel used will be value set with the CmdTestSetFrequency method. By default, this frequency will be the lowest one in the hop table, so the hop table can be defined as an alternative for setting this frequency explicitly.

To add support for CW/PRBS feature, five new parameters have been added in API. These parameters are set to turn on the CW/PRBS in CONTINUOUS or TIMED mode.
   a) TMR_PARAM_REGULATORY_MODE (Continuous or Timed)
   b) TMR_PARAM_REGULATORY_MODULATION (CW or PRBS)
   c) TMR_PARAM_REGULATORY_ONTIME (milliseconds)
   d) TMR_PARAM_REGULATORY_OFFTIME (milliseconds)
   e) TMR_PARAM_REGULATORY_ENABLE

Refer to “RegulatoryTesting” sample codelet in the MercuryAPI SDK to test this functionality.

**Support for Per-Antenna On-Time Settings**

With this Micro and Nano FW version, it is possible to schedule per antenna ON times in continuous read mode. Previously, this was only possible through API control, resulting in significant delay between one antenna turning off and the next turning on. Now the API can set “RF ON time” directly for each antenna at the module level, delegating all timing and synchronization issues to the module. As before, the antenna switching order is defined, along with the read time.

The default is to have no per-antenna on-time settings; i.e., use the dynamic search timeout behavior (switch when 5 inventory rounds result in no tags found). If any per-antenna on-times are set, then these override the default behavior.

There are no changes to the user-visible interface as the API already provided this feature through a weighted MultiReadPlan, but now the information is transferred to the module for it to execute. Weighted per-antenna reading is now executed as an asynchronous (continuous) read rather than a series of timed, synchronous, reads.

**Java Example:**

```java
// Create a multi read plan with 2 simple read plans.
```
//Set async on time to 1000ms.
r.paramSet(TMConstants.TMR_PARAM_READASYNCONTIME, 1000);

//Set async off time to 500ms.
r.paramSet(TMConstants.TMR_PARAM_READASYNOFFTIME, 500);

//multi read plan creation
SimpleReadPlan plan1 = new SimpleReadPlan(new int[]{1, 2}, TagProtocol.GEN2, null, null, 500);
SimpleReadPlan plan2 = new SimpleReadPlan(new int[]{2, 0, 1}, TagProtocol.GEN2, null, null, 1000);
MultiReadPlan plans = new MultiReadPlan(new ReadPlan[]{plan1, plan2});
r.paramSet(TMConstants.TMR_PARAM_READ_PLAN, plans);

So, as per the readplan weights and async on time set, the antenna search/read order will be 1 -> 2 -> 2 -> offtime(indicating 0) -> 1.

API will calculate how much read / on time is allocated to each antenna and the order to use antennas is sent as part of 0x91 command with 0x07 option.

<table>
<thead>
<tr>
<th>Readplan</th>
<th>weights</th>
<th>antennas in order</th>
</tr>
</thead>
<tbody>
<tr>
<td>plan1</td>
<td>500</td>
<td>1, 2</td>
</tr>
<tr>
<td>plan2</td>
<td>1000</td>
<td>2, 0, 1</td>
</tr>
</tbody>
</table>

Since async on time is set to 1000ms, and weight of plan 1 is 500, total on time allocated to plan 1 is calculated as below.

\[ \text{Asyncontime} \times \left(\frac{\text{weight of plan1}}{\text{total weight} (i.e., \text{weight of plan1} + \text{weight of plan 2})}\right) \]

i.e., \(1000 \times \left(\frac{500}{1500}\right) = 333\text{ms}\)

This 333ms is allocated to plan1. But plan 1 contains 2 antennas \{1, 2\}, so each antenna will take \(333/2 = 166\text{ ms} (0xA6)\). Hence ant 1 takes 166 ms and then ant 2 takes 166 ms.

Similarly, if we consider plan2, it's weight is 1000. The read / on time allocated for plan2 will be \(1000 \times \left(\frac{1000}{1500}\right) = 666\text{ms}\)

This 666ms is allocated to plan2. But plan2 contains 2 antennas \{2, 1\}, so each antenna will take \(666/2 = 333\text{ ms} (0x14D)\). Hence antenna 2 takes 333ms and then executes off time(will calculate below) and then reads on antenna 1 for 333ms.

Off time calculation:

--------------------------
Off time is indicated using number '0' in read plan (only for multi read plan case, if used for simple readplan results in error stating invalid antenna number). Since '0' is used in plan 2 only, off time will be executed for plan 2 only.
AsyncOffTime *(weight of plan2 / total weight(i.e., weight of plan1 + weight of plan 2))

i.e, 500 * (1000 / 1500) = 333ms

Hence, read order should be as:
ant 1 -> 166 ms, ant 2 -> 166ms, ant 2 -> 333ms, offtime -> 333 ms, ant 1-> 333ms.

Once this order is completed, then it again executes in circular / round robin order repeating the same order until stop read is sent or the specified time elapses.

Support for 5 new regions
Support for Vietnam, Thailand, Argentina, Hong Kong and Bangladesh regions has been added for Nano module. So, 5 new enum numbers have been added to “TMR_Region” enum to support newly added regions through API.

/** Vietnam **/TMR_REGION_VN = 25,
/** Thailand **/TMR_REGION_TH = 26,
/** Argentina **/TMR_REGION_AR = 27,
/** Hong Kong **/TMR_REGION_HK = 28,
/** Bangladesh **/TMR_REGION_BD = 29,

Here is a table summarizing details of 5 new regions:

<table>
<thead>
<tr>
<th>Country</th>
<th>Vietnam (VN)</th>
<th>Thailand (TH)</th>
<th>Argentina (AR)</th>
<th>Hong Kong (HK)</th>
<th>Bangladesh (BD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Region code</td>
<td>0x19</td>
<td>0x1A</td>
<td>0x1B</td>
<td>0x1C</td>
<td>0x1D</td>
</tr>
<tr>
<td>Lowest freq permitte</td>
<td>866000</td>
<td>920000</td>
<td>915000</td>
<td>865000</td>
<td>925000</td>
</tr>
<tr>
<td>Highest freq permitte</td>
<td>869000</td>
<td>925000</td>
<td>928000</td>
<td>868000</td>
<td>927000</td>
</tr>
<tr>
<td>Smallest step size</td>
<td>50</td>
<td>250</td>
<td>250</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Maximum dwell time</td>
<td>0.4 sec</td>
<td>0.4 sec</td>
<td>0.4 sec</td>
<td>0.4 sec</td>
<td>0.4 sec</td>
</tr>
<tr>
<td>Max RF power</td>
<td>27 dBm</td>
<td>27 dBm</td>
<td>27 dBm</td>
<td>27 dBm</td>
<td>27 dBm</td>
</tr>
<tr>
<td>LBT level</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Lowest channel in hop table</td>
<td>866450</td>
<td>920750</td>
<td>915250</td>
<td>865100</td>
<td>925400</td>
</tr>
<tr>
<td>Highest channel in hop table</td>
<td>867250</td>
<td>924250</td>
<td>927250</td>
<td>867900</td>
<td>926600</td>
</tr>
</tbody>
</table>
New Features of Universal Reader Assistant

Antenna Multiplexing and GPIOs Implementation in URA

This implementation provides user to view the module GPIO pins list and also would create a provision for the user to configure GPIO pins to either “Antenna Multiplexing”, which multiplex existing physical antennas to logical antennas or to input/output direction and value to HIGH/LOW (if it is output pin).

A GPIO pin can’t be configured as both at a time. User can set a configured input/output GPIO pin to a multiplexer. But, if a GPIO pin is defined as being used to control a multiplexer, it can no longer be defined as an input or output pin (it will be an output controlled by firmware) nor can its state be manually controlled.

For the remaining pins, GPIO input and output pins can be demonstrated as follows. As an input:

- Read when GPI line is asserted (whether it reads for a fixed time or continuously until the line de-asserted will be determined by the settings in the “Read Behavior” panel

<table>
<thead>
<tr>
<th>Country</th>
<th>Vietnam (VN)</th>
<th>Thailand (TH)</th>
<th>Argentina (AR)</th>
<th>Hong Kong (HK)</th>
<th>Bangladesh (BD)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>86650</td>
<td>921250</td>
<td>921250</td>
<td></td>
<td>92600</td>
</tr>
<tr>
<td></td>
<td>866850</td>
<td>923750</td>
<td>923750</td>
<td></td>
<td>926200</td>
</tr>
<tr>
<td></td>
<td>867150</td>
<td>919250</td>
<td>919250</td>
<td></td>
<td>925800</td>
</tr>
<tr>
<td></td>
<td>866450</td>
<td>916750</td>
<td>916750</td>
<td></td>
<td>926600</td>
</tr>
<tr>
<td></td>
<td>866750</td>
<td>926250</td>
<td>926250</td>
<td></td>
<td>925400</td>
</tr>
<tr>
<td></td>
<td>867050</td>
<td>922750</td>
<td>922750</td>
<td></td>
<td>926400</td>
</tr>
<tr>
<td>Hop table</td>
<td>866950</td>
<td>915750</td>
<td>915750</td>
<td>866500</td>
<td>925600</td>
</tr>
<tr>
<td></td>
<td>867250</td>
<td>918250</td>
<td>918250</td>
<td>867100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>921250</td>
<td>915750</td>
<td>925250</td>
<td>865900</td>
<td></td>
</tr>
<tr>
<td></td>
<td>923250</td>
<td>924750</td>
<td>924750</td>
<td>866300</td>
<td></td>
</tr>
<tr>
<td></td>
<td>922250</td>
<td>920250</td>
<td>920250</td>
<td>865500</td>
<td></td>
</tr>
<tr>
<td></td>
<td>92450</td>
<td>917750</td>
<td>917750</td>
<td>866100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>921750</td>
<td>927250</td>
<td>927250</td>
<td>867900</td>
<td></td>
</tr>
<tr>
<td></td>
<td>923750</td>
<td>921750</td>
<td>921750</td>
<td>865300</td>
<td></td>
</tr>
<tr>
<td></td>
<td>920750</td>
<td>923250</td>
<td>923250</td>
<td>866700</td>
<td></td>
</tr>
<tr>
<td></td>
<td>922750</td>
<td>919750</td>
<td>919750</td>
<td>867300</td>
<td></td>
</tr>
<tr>
<td></td>
<td>921250</td>
<td>916250</td>
<td>916250</td>
<td>865100</td>
<td></td>
</tr>
<tr>
<td></td>
<td>923750</td>
<td>926750</td>
<td>926750</td>
<td>865900</td>
<td></td>
</tr>
<tr>
<td></td>
<td>922250</td>
<td>922250</td>
<td>922250</td>
<td>867500</td>
<td></td>
</tr>
<tr>
<td></td>
<td>918750</td>
<td>915250</td>
<td>915250</td>
<td>865700</td>
<td></td>
</tr>
<tr>
<td></td>
<td>917250</td>
<td>925750</td>
<td>925750</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>924250</td>
<td>924250</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>920750</td>
<td>920750</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>917250</td>
<td>917250</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
• Whether asserted when low or high should be configurable (as we do in Autoconfig Tool and RAINstream)

As an output: Indicate read events. The duration that the output is asserted.

Number of GPIOs and antenna mux GPO pins are dependent on the type of reader (module) connected.

### GPIOs Summary:

<table>
<thead>
<tr>
<th>Module</th>
<th>GPIOs</th>
<th>PortSwitchGPOs</th>
<th>No. of logical antennas</th>
</tr>
</thead>
<tbody>
<tr>
<td>M6e</td>
<td>4</td>
<td>3</td>
<td>32</td>
</tr>
<tr>
<td>Micro</td>
<td>2</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>Nano</td>
<td>4</td>
<td>4</td>
<td>16</td>
</tr>
</tbody>
</table>

### GPIO State to Logical Antenna Mapping

The following charts give the complete mapping of GPO states to logical antenna numbers for the M6e, Micro, and Nano modules. Also shown is the physical antenna on the module which is active and presumably fed into a multiplexer controlled by the GPO lines.

If any GPO line is unused, assume its state is permanently low and eliminate all row entries corresponding to a high state for that GPO line – those logical antenna numbers will not be used. This method has the following advantages:

1. The low number logical antennas always correspond to the physical ports, for example, logical port 4 will always correspond to physical port 4 on the module regardless of the use of GPO lines.
2. Use or non-use of GPO lines will result in logical antennas being added or deleted, but the port numbering will always remain consistent. For example, logical port 6 corresponds to GPO1 low and GPO2 high. If GPO3 is added to control a multiplexer, port 6 will still correspond to GPO1 low and GPO2 high if GPO3 is low. There will not be a complete re-association of GPO states to logical antenna numbers.

#### M6e Logical Antennas Mapping:

<table>
<thead>
<tr>
<th>Logical Antenna</th>
<th>GPO 1 State</th>
<th>GPO 2 State</th>
<th>GPO 3 State</th>
<th>Active M6e Physical Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>J1</td>
</tr>
<tr>
<td>Logical Antenna</td>
<td>GPO 1 State</td>
<td>GPO 2 State</td>
<td>GPO 3 State</td>
<td>Active M6e Physical Port</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------</td>
<td>-------------</td>
<td>-------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>2</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>J2</td>
</tr>
<tr>
<td>3</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>J3</td>
</tr>
<tr>
<td>4</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>J4</td>
</tr>
<tr>
<td>5</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
<td>J1</td>
</tr>
<tr>
<td>6</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
<td>J2</td>
</tr>
<tr>
<td>7</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
<td>J3</td>
</tr>
<tr>
<td>8</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
<td>J4</td>
</tr>
<tr>
<td>9</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
<td>J1</td>
</tr>
<tr>
<td>10</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
<td>J2</td>
</tr>
<tr>
<td>11</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
<td>J3</td>
</tr>
<tr>
<td>12</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
<td>J4</td>
</tr>
<tr>
<td>13</td>
<td>High</td>
<td>High</td>
<td>Low</td>
<td>J1</td>
</tr>
<tr>
<td>14</td>
<td>High</td>
<td>High</td>
<td>Low</td>
<td>J2</td>
</tr>
<tr>
<td>15</td>
<td>High</td>
<td>High</td>
<td>Low</td>
<td>J3</td>
</tr>
<tr>
<td>16</td>
<td>High</td>
<td>High</td>
<td>Low</td>
<td>J4</td>
</tr>
<tr>
<td>17</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>J1</td>
</tr>
<tr>
<td>18</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>J2</td>
</tr>
<tr>
<td>19</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>J3</td>
</tr>
<tr>
<td>20</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>J4</td>
</tr>
<tr>
<td>Logical Antenna</td>
<td>GPO 1 State</td>
<td>GPO 2 State</td>
<td>GPO 3 State</td>
<td>Active M6e Physical Port</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------</td>
<td>-------------</td>
<td>-------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>21</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>J1</td>
</tr>
<tr>
<td>22</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>J2</td>
</tr>
<tr>
<td>23</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>J3</td>
</tr>
<tr>
<td>24</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>J4</td>
</tr>
<tr>
<td>25</td>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>J1</td>
</tr>
<tr>
<td>26</td>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>J2</td>
</tr>
<tr>
<td>27</td>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>J3</td>
</tr>
<tr>
<td>28</td>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>J4</td>
</tr>
<tr>
<td>29</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>J1</td>
</tr>
<tr>
<td>30</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>J2</td>
</tr>
<tr>
<td>31</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>J3</td>
</tr>
<tr>
<td>32</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>J4</td>
</tr>
</tbody>
</table>

**Micro Logical Antennas Mapping:**

<table>
<thead>
<tr>
<th>Logical Antenna</th>
<th>GPO 1 State</th>
<th>GPO 2 State</th>
<th>Active Micro Physical Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Low</td>
<td>Low</td>
<td>J1</td>
</tr>
<tr>
<td>2</td>
<td>Low</td>
<td>Low</td>
<td>J2</td>
</tr>
<tr>
<td>3</td>
<td>Low</td>
<td>High</td>
<td>J1</td>
</tr>
<tr>
<td>4</td>
<td>Low</td>
<td>High</td>
<td>J2</td>
</tr>
<tr>
<td></td>
<td>Logical Antenna</td>
<td>GPO 1 State</td>
<td>GPO 2 State</td>
</tr>
<tr>
<td>---</td>
<td>-----------------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>5</td>
<td>High</td>
<td>Low</td>
<td>J1</td>
</tr>
<tr>
<td>6</td>
<td>High</td>
<td>Low</td>
<td>J2</td>
</tr>
<tr>
<td>7</td>
<td>High</td>
<td>High</td>
<td>J1</td>
</tr>
<tr>
<td>8</td>
<td>High</td>
<td>High</td>
<td>J2</td>
</tr>
</tbody>
</table>

**Nano Logical Antennas Mapping:**

<table>
<thead>
<tr>
<th>Logical Antenna</th>
<th>GPO 1 State</th>
<th>GPO 2 State</th>
<th>GPO 3 State</th>
<th>GPO 4 State</th>
<th>Active M6e Physical Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>J1</td>
</tr>
<tr>
<td>2</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>J1</td>
</tr>
<tr>
<td>3</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
<td>J1</td>
</tr>
<tr>
<td>4</td>
<td>High</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
<td>J1</td>
</tr>
<tr>
<td>5</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
<td>J1</td>
</tr>
<tr>
<td>6</td>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
<td>J1</td>
</tr>
<tr>
<td>7</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>Low</td>
<td>J1</td>
</tr>
<tr>
<td>8</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>Low</td>
<td>J1</td>
</tr>
<tr>
<td>9</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>J1</td>
</tr>
<tr>
<td>10</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>J1</td>
</tr>
<tr>
<td>11</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>J1</td>
</tr>
<tr>
<td>12</td>
<td>High</td>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>J1</td>
</tr>
<tr>
<td>13</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>J1</td>
</tr>
<tr>
<td>14</td>
<td>High</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>J1</td>
</tr>
<tr>
<td>Logical Antenna</td>
<td>GPO 1 State</td>
<td>GPO 2 State</td>
<td>GPO 3 State</td>
<td>GPO 4 State</td>
<td>Active M6e Physical Port</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------</td>
<td>-------------</td>
<td>-------------</td>
<td>-------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>15</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>J1</td>
</tr>
<tr>
<td>16</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>J1</td>
</tr>
</tbody>
</table>
The below screenshots are from URA operated with M6e Reader.

1) The default Read/Write Expander Design is as below. ‘Antenna multiplexing’ checkbox is added additionally to create a provision for the user to multiplex into Logical Antennas. Additionally, ‘GPIO Configuration’ group box is added for configuring input/output pins.

![Screenshot of Read/Write Options](image-url)
2) When 'Antenna multiplexing' checkbox is checked, GPIO pins would be populated based on the module.

![Read/Write Options Diagram]

- **Antennas**: 01, 02, 03, 04
  - **Antenna Detection**
  - **Antenna Multiplexing**
- **Switching Method**:
  - Equal Time
  - Dynamic
- **GPIO Configuration**:
  - GPIO1
  - GPIO2
  - GPIO3
  - GPIO4
- **Read Behaviour**:
  - Read Continuously
  - RF On (ms): 1000
  - RF Off (ms): 0
- **Protocols**:
  - Gen2
  - ISO18000-6B
  - IPX64
  - IPX256
  - ATA
- **Embedded ReadData**
  - Enable
- **Filter**
  - Apply Filter
3) When ‘GPO1’ checkbox is checked, the logical antennas would be populated on the left of Antennas group box based on the module (reader) Connected. Logical Antenna’s list based on the module connected and number of GPO pins selected are mentioned in the above Section.

Note: - GPO1 pin is disabled under “GPIO Configuration” group box.
4) When all the GPIO pins are checked, all the logical antennas would be populated and the GPIO pins are disabled in GPIO Configuration Section. GPIO pins in antenna multiplexing and GPIO configuration are mutually exclusive, as when one is selected, other is disabled and vice-versa.

Note: - GPIO1, GPIO2, GPIO3 pins under “GPIO Configuration” group box section is disabled.
5) When a GPIO pin is checked, the behavior and logic level of that particular pin is popped up, and provision is created for the user to configure it for output.
6) When the 'output' radio button is checked, user could configure that particular GPIO pin to low/high.
7) When all the GPIO pins are checked and if the pins are already set to 'input', then Logic level would be populated for each of the pins.

8) When the user wants the GPIO pin to be multiplexed, then the screen would be as above.
Note: UI is designed with antenna multiplexing as the top priority. Hence, GPIO configuration is suggested only when that particular pin is not multiplexed primarily. Additionally, if the GPIO pin is configured to input/output, then user can opt for multiplexing, but the reverse mechanism is not encouraged or recommended.

Note: This is showing a multiplexer configuration screen, but not general use of GPO and GPI. GPI is most commonly used to enable reading in Autonomous Mode. If timed reading was selected, the read would occur once and not occur again until the GPI line were de-asserted and asserted again. Which state was “on” would need to be configurable. GPO could be asserted whenever a selected tag (or any tag) was read. There would need to be a configurable on-time.
URA UI Changes

1) In wizard screen, the button content in the Connection wizard is changed from “Cancel” to “Skip”. Earlier, user must press “Cancel” if they want to jump to the main screen. This is confusing to users as they expect to exit backwards, not move forwards in the process.

![Universal Reader Assistant - Connection Wizard](image)

2) The content of Disconnect and Connect button is changed. Removed “...” from top button when it says ”Disconnect...” as this implies it will bring user to another screen, not execute”.

When button at the top changes to “Disconnect”, it still is followed by the ellipsis as if the user were going to be brought to the “Connect” panel as before but disconnect is immediate. For this reason, the ellipsis (“...”) should be removed when the button says “Disconnect”.

![Universal Reader Assistant Connection Wizard](image)
3) **Settings Scroll Viewer UI Changes:**

Settings Scroll viewer is auto-sized as per the content available in the pane. Its visibility could be hidden by using autohide option available at the top right corner, below ‘Jadaktech’ logo.

![Image of Settings Scroll Viewer UI](image)

4) **Read/Write Expander Design Change:**

Read/Write Expander is redesigned with Antennas group box followed by GPIO Configuration. Previously Read Behavior is placed at the top of the expander.

![Image of Read/Write Expander](image)
New Features of Autonomous Configuration Tool

No features have been implemented in Config Tool during this release except logo, and icon changes from existing ‘ThingMagic’ to ‘Jadak’.
Resolved Bugs

The following API, URA and ACT bugs, identified in the release notes for version 1.31.0, have been addressed in version 1.31.1:

**MercuryAPI**

**Negative timestamp issue:**
Negative timestamp issue between consecutive tag reads around RF cycle has been fixed. The tag read timestamp is now more accurate with the system time for async read. With the previous released API version, a negative time difference is observed between the "timestamp of last tag read" before getting 0x400 response and the "timestamp of first tag read" after 0x400 getting response. (Ref #5767).

**Mercury API changes made to fix RF gaps between two sub read plans:**
“1000+ms timestamp issue on antenna cycling” and “1ms timestamp issue on 2nd antenna” issues have been fixed by adding tag fetching mechanism into serialreader.c after every read subplan. With the previous released API version, 1000ms difference is observed between the timestamps of two tags whenever the read plan sequence is getting started and 1ms difference is observed between the time stamps of the two tags while reading with 2nd read plan antenna. (Ref #5604).

**Sargas connection issue with Java API for JP2 region:**
Now user will be able to connect Sargas reader with Java API when it has JP2 region on it. With previous API version v1.31.0, Java API fails to connect Sargas reader when it is set to JP2 region. (Ref #5835).

**Timeout issue with >5 sec off time:**
Timeout issue has been fixed by changing the CommandTimeOut equation from “transportTimeout+asyncOnTime” to “transportTimeout+asyncOnTime+asyncOffTime”. With the previous released API version, performing async read with >5000 asyncOffTime results in “Timeout” error. (Ref #5592).

**Autonomous Configuration Tool**
Now Autonomous Configuration Tool is saving the “WritePower(/reader/radio/writePower)” parameter to a file along with all other parameters, when user clicks on “Save” button under “Profile” pane. (Ref #5635).
Operational Notes

MercuryAPI
The following restrictions and caveats apply to the features and functionality of API version 1.31.1

- “Load Save Configuration” functionality doesn’t work with M6 reader. API will not save current reader configuration to a file with “LoadSaveConfiguration” sample codelet. (Ref #5972).

- Gen2 lock tag operation doesn’t work with Java API when access password is “0xFFFFFFFF”, user will get “Write Tag Failed(0x406h)” error message. (Ref #5971).

- Gen2 T4 parameter doesn’t work on serial readers with C# API. User will get “Error: Specified cast is not valid.” exception when they set Gen2 T4 value as UInt32 type and will get “Error: Wrong type Int32 for parameter ”/reader/gen2/t4” exception with Integer type. (Ref #5975).

Universal Reader Assistant
The following restrictions and caveats apply to the features and functionality of URA version 4.1.36

- User will not find newly added features information in URA Help file. Refer “New Features of Universal Reader Assistant” section to get the information about newly added URA features. (Ref #5967).

- If user changes multiple GPIO pins state from input to output-High, one after another, URA configures all selected GPIO pins to output direction but configures only last modified GPIO pin value to HIGH state. (Ref #5965).
MercuryAPI 1.31.0 Release Notes

Includes release notes for Universal Reader Assistant v3.5.33 and the Autonomous Configuration Tool, v1.2.1.4. This API release is fully qualified for the entire ThingMagic product line.

These release notes describe the features of MercuryAPI SDK version 1.31.0 relative to version 1.29.4. The release notes for previous versions, going back to 1.27.3 (the last time the MercuryAPI Programmers Guide was updated) are included.

Compatible Firmware

MercuryAPI version 1.31.0 has been designed to support all the features of the following firmware releases. (Some are in the final stages of testing and may not be uploaded to the Support web site yet).

- M6e module firmware version 1.21.2
- Micro/Micro-LTE modules, and USBPro reader, firmware version 1.9.1
- Nano module firmware version 1.7.3
- Sargas reader firmware version 5.3.0
- M6 and Astra-EX reader firmware version 4.19.3
- M5e/M5e-Compact modules, Vega reader, and USB Plus+ reader, firmware version 1.7.5

All features are not available for all products due to firmware or hardware limitations. These release notes will note which products support the new features. Information about support for existing features are found in the MercuryAPI Programmers Guide.
# Features Matrix

The following table explains on the list of the features implemented during this release and supported products for each feature.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Supported Modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Margin read support for Monza 6 tags</td>
<td>M6e, Micro, Nano</td>
</tr>
<tr>
<td>NXP UCODE data flip issue in EPC</td>
<td>M6e, Micro</td>
</tr>
<tr>
<td>Gen2 parameters in metadata</td>
<td>M6e, Micro, Nano</td>
</tr>
<tr>
<td>LTKC/LLRP Support on Windows Platform</td>
<td>All LLRP Readers*</td>
</tr>
<tr>
<td>Support for Gen2V2(NMV2D) tags</td>
<td>M6e, Micro, Nano, Sargas (New)</td>
</tr>
<tr>
<td>Support for Gen2V2 embedded tag ops</td>
<td>M6e, Micro, Sargas (New)</td>
</tr>
<tr>
<td>Configurable T4 for Gen2 protocol</td>
<td>M6e, Micro, Nano, Sargas (New)</td>
</tr>
<tr>
<td>Add option to read data as part of Write EPC or Write Data command</td>
<td>M6e, Micro, Nano, Sargas (New)</td>
</tr>
<tr>
<td>Persistent Antenna config after across tag read multiple command (0x22)</td>
<td>M6e, Micro, Nano</td>
</tr>
<tr>
<td>Support for new Asian regions</td>
<td>M6e, Micro, Nano</td>
</tr>
<tr>
<td>Support for set/get quantizer and min frequency</td>
<td>M6e, Micro, Nano</td>
</tr>
<tr>
<td>Adding missing config fields in Autonomous tool</td>
<td>M6e, Micro, Nano</td>
</tr>
<tr>
<td>Function/Parameter to disable License key</td>
<td>M6e, Micro</td>
</tr>
<tr>
<td>Get HopTable param support in LLRP API</td>
<td>All LLRP Readers*</td>
</tr>
</tbody>
</table>

*M6, Astra-EX, Sargas

# New Features of API

The new features of the API are described briefly below. API version v1.31.0 is now the recommended release for all ThingMagic readers users.
See the MercuryAPI Programmer’s Guide for more information:

**MercuryAPI 1.31.0**

As shown in the table above, Mercury API 1.31.0 adds features to Sargas which were previously available only on modules:

- Support for Gen2V2 Untraceable, Authenticate, and ReadBuffer commands, as stand-alone or tag operations. Currently only supported in NXP UCODE DNA tags and Acura NMV2D tags (special tags for toll collection)
  - Note: At this time, the wrong amount of data is sent for TAM2 authentication when SendRaw is “true” (Ref# 5701)
- Ability to vary the T4 timer for tags which need additional time between Select and Query, such as sensor tags which use Select to trigger reading of the sensor value and placing it in memory where it can be read.
- Ability to read data immediately after a write command. This is most useful when the write command is used by sensor tags to signal that the tag should read its sensor value and place it in memory, but can be used by any application which benefits from doing both these actions in succession.
- Support for Gen2 protocol extensions used in Brazil (Denatran). A license, installed through the API, is required to activate these features. The presence of the license can be detected by the API, but there is no indication in the web interface. (Ref# 5165)
- Reboot() function is now available through the API (previously, only the web interface could reboot the reader). (Ref# 5651)

**MercuryAPI 1.29.4**

**Margin read support for Monza 6 tags**

MarginRead is an EPC Gen2 compliant custom command supported by tag chips with Integra. This command allows a reader to explicitly verify that the non-volatile memory (NVM) in the tag chip is not weakly written, guaranteeing a minimum margin on NVM. It is used for quality control to ensure data integrity and for failure analysis.

There are several ways that the MarginRead command could be used with Monza 6. A recommended use of MarginRead is independent verification of the encoding quality, either on a sample basis or for diagnosis during failure analysis.

A basic understanding of MarginRead:

When data is written to a tag using the Gen2 protocol, charge is built up in the memory cells until they reach the appropriate level. Once that happens, the tag returns a "done" signal telling the interrogator (reader) or encoding system that the write operation has completed successfully.
It is a known field issue that not all encoding systems properly wait for the "done" signal and instead issue a read operation to check if the data is correct. A read operation may return correct data even if the write operation did not complete successfully.

A partially charged memory cell might retain data for a limited time but then it will lose data integrity over time. Data retention could be for an unpredictable amount of time from a few minutes to several years.

A fully charged memory cell will retain data for a long period of time. Specifically, the Monza 6 tag is expected to retain data for up to 50 years.

The MarginRead command allows customers to check if Monza 6 tag chip memory cells are fully charged.

If a customer encounters a data integrity issue in the field, MarginRead may be used for diagnostics. If MarginRead indicates an issue, then the encoding method should be investigated.

**C API:**
New chip type (TMR_SR_GEN2_IMPINJ_MONZA6_SILICON = 0x0D) and tagOp (TMR_TagOp_GEN2_Impinj_Monza6_MarginRead) is introduced in tmr_gen2.h and tmr_tagop.h respectively.

**C# API:**
New chip type (GEN2_IMPINJ_MONZA6_SILICON = 0x0D) and tagOp (MARGIN_READ = 0x00) is introduced in Gen2.cs.

**Java API:**
New chip type (GEN2_IMPINJ_MONZA6_SILICON = 0x0D) and tagOp (MARGIN_READ = 0x00) is introduced in Gen2.java and EmbeddedReaderMessage.java respectively.

**Configuration word change support for NXP UCODE 7 tag**

Prior to UCODE 7, NXP supported a set of custom commands that could change the configuration word values. Unfortunately, these commands that was developed for the G2i line of tags do not work for the UCODE 7 tags. A new custom command has been implemented in API to change NXP UCODE 7 configuration word on M6e/Micro.

For UCODE 7 tags, support for ChangeConfig command is dropped. An alternative way to change the configuration word for UCODE 7 tags is needed.

UCODE 7 configuration word contains 2 different type of bits:

- Action bits: meant to trigger a feature upon a SELECT command on the related bit:
  - Parallel encoding (at address 0x202)
  - Tag Power indicator (at address 0x204)
❖ Permanent bits: permanently stored bits in the memory
   Max. Backscatter Strength (at address 0x209)
   PSF Alarm bit (at address 0x20F)

**Java API:**
A new custom command to change NXP UCODE7 configuration word has been implemented in the API. API change includes addition of a new siliconType(chip type) named “GEN2_NXP_UCODE7_SILICON(0x0E)” and added subcommand option to change configword “UCODE7_SUBCOMMAND_CHANGE_CONFIG = 0x08;”. This custom tagop supports both standalone and embedded tag operation.

User can activate or de-activate the feature behind the action bits and permanent bits by setting the corresponding bit value in the configuration word.

A similar change has been made for the C# API.

**C API:**
A new custom command is added to configure NXP UCODE7 configuration word has been implemented in the API.

New chip type (TMR_SR_GEN2_NXP_UCODE7_SILICON = 0x0E) and tagOp (TMR_TagOp_GEN2_NXP_UCODE7_Config) is introduced in tmr_gen2.h and tmr_tagop.h respectively

**Gen2 parameters in metadata**
Now that we allow modification of the Gen2 parameters on the fly meaning when reader actively reading tags, it would be helpful to include current Gen2 settings as metadata in tag reads.

For example, the fact that Gen2 Q value can change dynamically and if the customer is trying to determine the best static value, it would be good to know whether our dynamic algorithm changes the Q often or seldom. Gen2 parameters that are included in metadata are:

❖ TAG_METADATA_GEN2_Q = 0x0200
❖ TAG_METADATA_GEN2_LF = 0x0400
❖ TAG_METADATA_GEN2_TARGET = 0x0800

User can fetch these flags(TagReadData.TagMetadataFlag.GEN2_Q, TagReadData.TagMetadataFlag.GEN2_LF, TagReadData.TagMetadataFlag.GEN2_TARGET) as part of metadata flags when requested before initiating read.

Read code sample demonstrates the addition of new metadata flags and prints these flags information to console when requested on M6e/Micro/Nano.
LTKC/LLRP Support on Windows Platform

We've only supported Windows CE/Mobile with C# before. Our C API does build on Windows, but it does not support LLRP, because LTKC (LLRP Toolkit - C) does not support Windows. However, there is LTKCPP, a C++ version of LTK, which does have Windows support. But we do not want to expand Mercury API support to yet another language. After initial discussion, finally we decided to create and build a project for LTKC in Window.

LLRP is the protocol that our intelligent fixed readers (Sargas, M6, Astra-EX) use to communicate with a host. In previous releases, we supported LLRP-based control programs, such as LLRP Commander, on Linux platforms. With this release, we also support LTKC, an open-source LLRP-based control program for Windows platforms. An application note is available from support@thingmagic.com which explains this more completely.

Support for Gen2V2(NMV2D) tags

In previous API releases, there is no flexibility of changing protMode values for the user. Since protMode = 0x01 is the only supported value for NXPUCODE AES tag and TAM2 response is always 256 bits. NMV2D tag support has been added in current API release, the change in the expected number of bits in TAM2 reply based on the protMode values. For protModes = 0x00, 0x01, the TAM2 reply consists of 256 bits whereas for protModes = 0x02, 0x03, the reply consists of 352 bits(16 bytes of Authentication block + 16 bytes of Data block + 12 bytes of CMAC block).

Refer to Authenticate, ReadBuffer and Untraceable sample codelets in the MercuryAPI SDK to test this functionality. Following API changes have been done to activate this functionality.

- API interface change to Tam2Authentication initialization method
- Codelet changes
  - Readbuffer codelet modification to use modified Tam2Authentication initialization method and bitCount support of 256 for protModes = 0 and 1, 352 for protModes= 2 & 3.
  - Authenticate codelet modification to use modified Tam2Authentication initialization method

Support for GEN2V2 Embedded tag ops

Embedded tag operations support has been added for both NXP UCODE DNA tag and NMV2D tags in all the 3 API’s. Please refer Authenticate, ReadBuffer and Untraceable codelets in each API for changes. This feature allows for high speed secure reading in Asynchronous modes.

Mercury API changes includes addition of 3 tagop cases for embedded tag operation of Authenticate, ReadBuffer and Untraceable. This feature supports on M6e and Micro readers.

Configurable T4 for Gen2 protocol

Some sensor tags use a Select command to trigger reading of their sensor. We found that the time our reader was waiting between the Select command and start of inventory (when the
reader sends the Query command) was insufficient for the sensor tag to obtain its reading before having to report it to the reader. This delay time is controlled by a Gen2 parameter called the “T4 timer”. So, we have added the ability to set the T4 timer to a larger value. For that a new parameter “/reader/gen2/t4” has been added in all the 3 API’s in order to set /get T4 parameter value on M6e/Micro/Nano.

**Param:** “TMR_PARAM_GEN2_T4” or “/reader/gen2/t4”  
**Type:** Integer

T4 is a integer value and specified in microseconds. Minimum value of T4 allowed is 64 microseconds. Max value allowed is 1 second. Here is an oscilloscope trace of the reader output signal showing the effect of changing this setting.

In the above image, 69.88ms indicates T4 time.
Read Data as part of Write EPC or Write Data command

Some sensor tags require the module to write to a memory bank to trigger the sensor measurement, then read the sensor data field without dropping power between (as happens if the two operations are done as separate commands). This functionality could be useful for streamlining read-then-write operations for other applications as well.

To support this requirement, Read data support has been added as an option for the Write EPC and Write bank data commands on M6e/Micro/Nano. This allows the module to read the data from any of the memory banks following a successful write operation of data to any memory bank (or write EPC) through a single command. The standard commands to Write Tag Data and Write Tag EPC will now optionally include the read memory bank, read word address, and read count to implement this feature..

For more details on the application interface, please refer WriteTag sample codelet in each API for changes and test this functionality.

Decoupling antenna selection from AsyncOnTime

In previous releases, when reading continuously, the reader would always return to antenna 1 (or the first antenna in the configured list) at the beginning of each AsyncOnTime cycle. This encouraged users to configure a high value for AsyncOnTime to ensure that all antennas would be activated each read cycle. However, some of the settings that can now be changed without interrupting reading will take effect only at the beginning of the next AsyncOnTime cycle, so users now have a contrary reason to set this value as small as possible.

To eliminate this contradiction, the antenna selection algorithm has been changed to remember the last antenna that was active in the previous read cycle and start with that antenna for the next AsyncOnTime cycle. This way, the active antenna cycles through the list with regularity and the AsyncOnTime can be optimized to make the time for on-the-fly settings take effect as quickly as possible, or for other reasons.

The ReadAsync code sample can be run to see the effect of this change. No API modifications has been done for this feature.
Support for New Asian regions

One of our customers calculated the optimum channel frequencies to permit the greatest number of channels while still meeting out-of-band emissions standards for Asian regions and Russia. Although the frequency range of these new regions was within the existing regions, the exact desired channels could not be realized with our quantization rules (all channels must be an even multiple of the quantization value above the low channel boundary for that region). To achieve the desired channels, the following additional regions have been added in M6e/Micro/Nano.

Regions that are added in current firmware version have the following characteristics.

<table>
<thead>
<tr>
<th>Region</th>
<th>Region Abbrev.</th>
<th>Region Number</th>
<th>Low Channel Boundary</th>
<th>High Channel Boundary</th>
<th>Min Step Size (Quantization)</th>
<th>Hop Table</th>
<th>Max RF Power Allowed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Malaysia</td>
<td>MY</td>
<td>0x10</td>
<td>919 MHz</td>
<td>923 MHz</td>
<td>250 kHz</td>
<td>921750, 919250, 920750, 922250, 919750, 921250, 920250, 922750</td>
<td>31.5 dBm</td>
</tr>
<tr>
<td>Indonesia</td>
<td>ID</td>
<td>0x11</td>
<td>923 MHz</td>
<td>925 MHz</td>
<td>125 kHz</td>
<td>924625, 923375, 924125, 923875, 924375, 923625, 924875, 923125</td>
<td>31.5 dBm</td>
</tr>
<tr>
<td>Philippines</td>
<td>PH</td>
<td>0x12</td>
<td>918 MHz</td>
<td>920 MHz</td>
<td>250 kHz</td>
<td>919250, 918750, 919750, 918250</td>
<td>31.5 dBm</td>
</tr>
<tr>
<td>Taiwan</td>
<td>TW</td>
<td>0x13</td>
<td>922 MHz</td>
<td>928 MHz</td>
<td>250 KHz</td>
<td>926250, 924750, 922250, 925750, 923250, 927750, 926750, 924250, 922750, 925250, 923750, 927250</td>
<td>30 dBm</td>
</tr>
<tr>
<td>Region</td>
<td>Region Abbrev.</td>
<td>Region Number</td>
<td>Low Channel Boundary</td>
<td>High Channel Boundary</td>
<td>Min Step Size (Quantization)</td>
<td>Hop Table</td>
<td>Max RF Power Allowed</td>
</tr>
<tr>
<td>--------</td>
<td>---------------</td>
<td>---------------</td>
<td>----------------------</td>
<td>-----------------------</td>
<td>-----------------------------</td>
<td>-----------</td>
<td>---------------------</td>
</tr>
<tr>
<td>Macao</td>
<td>MO</td>
<td>0x14</td>
<td>920 MHz</td>
<td>925 MHz</td>
<td>250 kHz</td>
<td>923250, 921750, 924250, 922750, 920250, 923750, 921250, 924750, 922250, 920750</td>
<td>31.5 dBm</td>
</tr>
<tr>
<td>Russia</td>
<td>RU</td>
<td>0x15</td>
<td>866 MHz</td>
<td>868 MHz</td>
<td>200 kHz</td>
<td>866600, 867800, 866200, 867000, 866400, 867600, 866800, 867200</td>
<td>31.5 dBm</td>
</tr>
<tr>
<td>Singapore</td>
<td>SG</td>
<td>0x16</td>
<td>920 MHz</td>
<td>925 MHz</td>
<td>100 kHz</td>
<td>923100, 921900, 924300, 920700, 922500, 923700, 921300, 924900, 920100</td>
<td>31.5 dBm</td>
</tr>
</tbody>
</table>

Notes:
(1) Maximum Dwell Time 0.4 sec for all these regions (same as North American region)
(2) Max RF power limit is that given in table or whatever the module is capable of, whichever is lower.
(3) Any channel frequency can be requested that is between the upper and lower bounds, but the module will silently round down to the nearest channel that is the lower bound plus an integer multiple of quantization steps.

The new Asian regions have been added to Reader.Region method. Use this API method to set/get above added Asian regions and test the functionality.

**Support for set/get quantizer and min frequency**

The Open region as defined in previous releases was intended for testing only. In order to permit the most flexibility in defining channels, it allowed a minimum channel step size (quantization) of 25 kHz. We did not recommend the use of the Open region to support channel plans which could not be easily accommodated by changing the hop table of an existing region because such a small step size will result in lower channel frequency stability.
To allow the Open region to be used more flexibility, quantization step value and minimum frequency value have been added in M6e, Micro and Nano FWs and to set/get quantization step value and minimum frequency value in API, two new parameters have been added in all the 3 API's.

**Param 1:** “TMR_PARAM_REGION_QUANTIZATION_STEP” or
”/reader/region/quantizationstep”
**Type:** uint32_t

**Param 2:** “TMR_PARAM_REGION_MINIMUM_FREQUENCY” or
”/reader/region/minimumfrequency”
**Type:** uint32_t

These values can only be set for OPEN region. Error will be returned when set in other regions. Get command returns the step and min freq of the current region. Input step value must be greater than 15KHz and less than 6MHz. These boundary values cannot be set. Error will be returned. Step value to be set should be such that the remainder of 6MHz and the step value must be 0 (i.e. 6MHz % step value == 0).
Function/Param to disable License key

In previous releases, we have only set **License key** param support and there is no option to disable any licensed feature using a License key parameter. In current release, erase license key support has been added in all the 3 API's.

**C API:**
In C API we have two basic structure which are related to license key.

a. TMR_SR_SetProtocolLicenseOption option; //contains option like set/erase.
   b. TMR_uint8List *license; //used to contain license key

**Java API:**
A new parameter named “/reader/manageLicenseKey” is added to the API. This param can be used to set/erase license key on the reader. Both set and erase license operations are supported for serial reader. But, network reader supports only "set" license key operation. If user tries to erase license key, he will be prompted with an error message “Unimplemented feature.”

Param: /reader/manageLicenseKey
Type: LicenseOperation

**C# API:**
A new parameter named “/reader/manageLicenseKey” is added to the API. This param can be used to set/erase license key on the reader. Both set and erase license operations are supported for serial reader. But, network reader supports only "set" license key operation. If user tries to erase license key, he will be prompted with an error message “Unimplemented feature.”

Param: /reader/manageLicenseKey
Type: LicenseOperation
Get Hoptable param support in LLRP API
In previous releases, Hoptable and Hoptime parameters are not supported in LLRP API as these are not configuration parameters as per LLRP standard. Get Hoptable support has been added in current API release and code has been modified to support this param as read only.

Nothing has been modified in module FW and finished reader.

New Features of URA
No Features implemented in URA during this release.

New Feature of Autonomous Configuration Tool

Version 1.2.1 derived from API 1.29.4

Adding Missing Configuration fields in Autonomous Configuration Tool
In previous API release, few persistent configuration parameters have been added in API but not in Autonomous Configuration Tool. In current release, those parameters have been included in Autonomous configuration tool for persistent configuration. The following are the params being added to ACT tool.

1. Frequency hop table and hop time.
2. Gen2 Tari and Gen2 Q.

This feature also supports Save and Load Configuration functionality for hoptable, hoptime and Gen2 Q, Tari parameters.

User can save these configurations with Gen2 Tari and Q and frequency hoptable and hoptime and restore them accordingly. The new ACT tool provides configuring these fields. Below screenshots will provide the UI look of new configurations in Autonomous Configuration Tool.
Frequency HopTable and HopTime Screenshot:

Gen2 Tari and Q Screenshot:
Bug Fixes

The following API, URA and Configuration Tool bugs, identified in the release notes for version 1.29.3, have been addressed in version 1.29.4:

**MercuryAPI 1.29.4**

- The firmwares released with current API version 1.29.4 has fix for RF AsyncOffTime issue and works fine up to 65535(0xFFFF) async off time value (The firmware released with API version 1.29.2 does not tolerate extremely long Async RF Off times and stopped responding to user requests until unless reader reboot. The limit for the M6e and Micro is 11454. The limit for the Nano is 4295). (Ref# 5154, 5157, 5159).

- Now performing sync read, after pseudo async read, working as expected.(In previous API release, performing sync read after pseudo async read was sending 2f command to the module instead of 22 command due to "useStreaming" flag is not made false whenever stopReading() is called). (Ref#5457).

- Embedded tag operations with filter now works fine with C API(With previous releases, C API returns ‘Message command length is incorrect’ error message when we perform embedded tag operations on M6/Micro/Sargas by enabling filter). (Ref# 5016).

- Now Java API will throw ‘operation not supported’ error message when user tries to read more than one gen2 bank data at a time on LLRP readers(With previous releases, “java.lang.NullPointerException” error message returned to the user, when he tries to read more than one bank data at a time on LLRP readers). (Ref# 5014).

**Universal Reader Assistant 3.4**

- Now URA “Equal Time” option is enabled, as an antenna switching method, for USBPro reader.(In previous releases, “Equal Time’ option as an antenna switching method, was in disable mode, when USBPro reader connected using URA). (Ref# 5160).

- Opening URA after PC restart/shutdown now works fine, doesn’t hang(With previous releases, URA hangs when user tries to open URA after PC restart/shutdown). (Ref #5417).

- Now HopTable and HopTime values are persistent in URA(In previous release, reconnecting the reader through URA always sets default HopTable and HopTable due to URA always setting the set region command at every connect time). (Ref# 5513).
Operational Notes

MercuryAPI 1.29.4

The following restrictions and caveats apply to the features and functionality of API version 1.29.4

- When the user does not set GPIO metadata flag and uses logical antenna extension, incorrect antenna id is displayed (for example:- If the module has logical antenna support of 1 to 16 and if user does not set GPIO metadata flag, user will get antenna id 17 to 32 instead of 1 to 16).(Ref # 5150).

- Android application may hangs, when doing start and stop read in a while loop.(Ref#5428)

Universal Reader Assistant 3.4

URA may throw “Unable to connect to Reader(COMxx). Please check if the device is properly connected or Device might be in use” error message, when user try to connect the reader in URA after restart/shutdown the PC during read is in progress. (Ref# 5466)
MercuryAPI 1.29.3 Release Notes

These release notes describe the features of MercuryAPI SDK version 1.29.3 relative to previous versions, going back to 1.27.3, the last time the MercuryAPI Programmers Guide was updated.

API release notes include those for Universal Reader Assistant v3.3.24 and the Autonomous Configuration Tool, v1.2.1. This API release is fully qualified for the entire ThingMagic product line.

Compatible Firmware

MercuryAPI version 1.29.3 has been designed to support all the features of the following firmware releases:

- M6e module firmware version 1.21.1
- Micro/Micro-LTE modules, and USBPro reader, firmware version 1.9.0
- Nano module firmware version 1.7.1
- Sargas reader firmware version 5.1.4
- M6 and Astra-EX reader firmware version 4.19.3
- M5e/M5e-Compact modules, Vega reader, and USB Plus+ reader, firmware version 1.7.2

All features are not available for all products due to firmware or hardware limitations. These release notes will note which products support the new features. Information about support for existing features are found in the MercuryAPI Programmers Guide.
New Features of the API

The new features of the API are described briefly below. All features previously described for interim MercuryAPI versions 1.29.0 and 1.29.1 for Sargas only are now combined into 1.29.2. This is for the convenience of non-Sargas users who have not used these interim versions, and also to reinforce that 1.29.2 is now the recommended release for all Sargas reader users.

See the MercuryAPI Programmers Guide for more information:

MercuryAPI 1.29.3

Additional Support for Processors without an OS

In previous release we only supported readasync on “bare metal processors” (those without an OS) for an ATML board (samd21j18a). With this release we have added the support for async reading on bare metal for STM32 board.

There are no changes in the asynch reading codelet itself. We have modified the serial transport to support the async read operation on this new platform. An application note is available from support@thingmagic.com which explains this more completely.

LTKC/LLRP Support on Windows Platforms

LLRP is the protocol that our intelligent fixed readers (Sargas, M6, Astra-EX) use to communicate with a host. In previous releases, we supported LLRP-based control programs, such as LLRP Commander, on Linux platforms. With this release, we also support LTKC, an open-source LLRP-based control program for Windows platforms. An application note is available from support@thingmagic.com which explains this more completely.

C++ WINCE Support

We have created a library project similar to C MercuryAPI which will support the WinCE platform, along with two sample applications:

- Sample_WinCE
- ReadAsync

These applications are replicas of the standard sync and async read apps. The only enhancement to their functionality is that reading can be started using GPI pins.
Changes to Code Samples (“Codelets”)

- The LicenseKey code samples (in all three APIs) has been enhanced to take the license key as a command-line option. For example:

  “tmr://COM4 --key AB CD EF {etc}"

  This eliminates the need to edit the source file and recompile it in order to use this utility to install a license key into a ThingMagic product.

- The FastID code sample in the Java API now sets the read and write power to the maximum value.

MercuryAPI 1.29.2

MercuryAPI 1.29.2 has added the following new features and functionality:

Sargas Reader Support

Support for the new Sargas reader was added in Mercury API version 1.29.0, and enhanced for version 1.29.1 and this version, 1.29.2. The code samples have been updated to support the Sargas reader.

AEI ATA and IP-X Protocol Support

Support for additional non-Gen2 protocols have been added to the following products:

- **Sargas Reader**: The API can now support the optional AEI ATA and IP-X protocols
- **M6e and Micro/Micro-LTE Readers**: The API can now support the optional AEI ATA protocol
- **Nano module, and all products derived from the M5e or M5e-Compact modules**: Only the Gen2 protocol is supported.

An AEI ATA decoding code sample has been added to illustrate how this functionality is accessed (“AEITagDecoding”).

Support for M6e-JIC module

Support for the new M6e-JIC module has been added to API. It supports China, Japan, and Israeli regions and will eventually replace the M6e-PRC module, which supported China and Japan only. Code samples have been updated to support this module. The new Israeli region is now recognized by the Mercury API.
Support for Processors with no OS

Documentation and examples have been provided for using the “C” API with small processors which have limited memory and no Operating System. The code sample “readasync_baremetal.c” was added to the SDK to illustrate how to use the C-API on a processor with no inherent ability to run separate code threads.

Support for On-Module Duty Cycle control

The API now supports control of the transmission duty cycle at the module level. Previous releases allowed the user to control duty cycle at the API level only - The module had to be told when to read and when to stop. Duty cycle was not supported at all under continuous reading. Now a duty cycle setting has been added to the module firmware so that users can reduce transmit on-time during autonomous operation in order to save battery life and limit the temperature rise of the module.

This feature is supported on the Nano, Micro/Micro-LTE, and M6e modules, and the USBPro reader only. It is not supported on the M5e/M5e-Compact modules nor on readers derived from these modules. The M6, Astra-EX, and Sargas readers will support duty cycle control as before - the host controlling these readers will direct the module to use the desired duty cycle and the reader will remember that setting until the reader is rebooted, but this setting will not be remembered across reboots.

Support for Configuring Which Meta-data is Returned by Modules

Previous versions of the API allowed the user to select which meta-data values (antenna, frequency, RSSI, protocol, etc.) was returned with every tag read report. In the background, however, the module was returning all possible values to the API and the API was discarding any the user did not want to see. Now, additional controls have been added to instruct the module not to collect the undesired data at all and not send it over its interface with the host. For some applications, where the speed of the interface to the module is limited, this can result in higher performance and fewer instances of data backing up in the module because it cannot be transferred to the host.

This feature is supported in the Nano, Micro/Micro-LTE, and M6e modules, as well as the USB Pro reader. It is not supported in the M5e/M5e-Compact family of readers or readers derived from these modules. It is not supported in the Sargas, M6 or Astra-EX readers.

Support for RAIN Gen2 EPC Truncation

The RAIN Gen2 standard allows for tag to report a subset of their EPC value if a significant portion of it is identical from tag to tag. This is called “EPC Truncation” and is
supported on the Nano, Micro/Micro-LTE, and M6e modules, as well as the USB Pro reader. It is currently not supported on any other reader or the M5e/M5e-Compact family of modules.

Filtering based on EPC Length
In order to eliminate reporting of stray and phantom tags, we have added the ability for the M6e, Micro/Micro-LTE, and Nano modules to only report tags of a desired length. This feature is also supported on the USB Pro reader, but not for any other readers, nor for the M5e/M5e-Compact series of modules.

Multiplexing Extension for M6e Module
The M6e and Micro/Micro-LTE modules have always had the ability to use two GPO lines to control a multiplexer that expands each port to 4 ports, with full functionality as if the ports were physically part of the module. With its latest release of firmware, the M6e module now can use 3 GPO lines to expand its multiplexing to expand each port to 8 ports. This changes extends the maximum number of “logical” ports to 32.

Only the M6e module supports this feature. The other modules and readers do not. (The Micro module only has 2 GPIO lines, so will never be able to support this feature.)

Enhanced Licensing Tool Support
The M6e and Micro/Micro-LTE, and Nano modules, as well as the readers derived from these modules, support protocols and features that are activated by installation of a license key. Previous versions of firmware required that a software utility be generated for each module or license purchased, which would only install the license if the module had the correct serial number. A new class of licenses has been defined that incorporates the serial number into the license, so that the module (or reader) firmware can do the checking. This freed us to create license installation utilities that can be used with any module or reader. This new licensing mechanism is supported on the modules previously mentioned and the Sargas and USB Pro readers. It is not yet supported on the M6 and Astra-EX readers. The M5e/M5e-Compact modules and the readers that are derived from them do not support licensed features at all.

Changing Settings During Continuous Reading
The M6e, Micro/Micro-LTE, and Nano modules support continuous reading. In this mode, a read plan is sent to the module and it executes that plan until told to stop. Until this release of firmware and API, it was necessary to stop continuous reading to alter any settings. With this release, a subset of RAIN Gen2 settings may be altered without stopping continuous reading. The supported settings are:
New Features of the API

- Global Read TX Power (not per-port power)
- Global Write TX Power (not per-port power)
- Gen2 BLF (250 or 640 kHz)
- Gen 2 TARI (6.24, 12.5, or 25 usec)
- Gen2 Encoding (“FM0” or “M”=2, 4, OR 8)
- Gen2 Q (“Dynamic” or a static value from 0 to 16)
- Gen2 Session (S0, S1, S2, or S3)
- Gen2 Target (A, B, AB, or BA)
- GPO set and GPI get (but not the control to change an input to an output or vice versa)

Note: Duty Cycle cannot be adjusted without turning off reading. Earlier editions of these release notes erroneously claimed it could.

Ability to Determine Module’s Ability to Support Custom Gen2 Capabilities

The host has the ability to “get” many settings when it first connects to a module in order to determine whether the settings are as desired or if they need to be changed. Now, an additional “get” is allowed to determine if any optional Gen2 extensions have been enabled for the module. (Currently there is only one custom extension offered - IAV Denatran support.) This feature is currently supported on the m6e and Micro modules as well as the USB Pro reader.

The parameter to get protocol extensions is /reader/Gen2/ProtocolExtension.

New and Modified Code Samples

The following changes and additions have been made to our code samples to illustrate the new features:

AEITagDecoding

- New sample to illustrate interpreting data from an AEI ATA tag data string. (C#/.NET and Java APIs only, not supported on the “C” API, yet.)

AutonomousMode

- Support for M6e-JIC Module Added

EmbeddedReadTID

- Support for M6e-JIC Module Added
- Support for Sargas Reader Added
Filter

- Modified to demonstrate tag filtering based on length and Gen2 Truncate filtering.

Gen2ReadAllMemoryBanks

- Support for M6e-JIC Module Added
- Support for Sargas Reader Added

GpioCommand

- Added support for Sargas, M6 and Astra-EX readers

Read

- Modified to illustrate limiting the collection of meta-data at the module level.
- Read.c now supports “bare metal” processors such as the STM32 and Atmel processors.

ReadAsync_baremetal

Demonstrates continuous reading using a processor without thread support. (“C” API only.)

ReadAsyncFilterISO18k6b

- Now contains an example of tag operations for ISO 18000-6B protocol

SavedConfig

- Support for M6e-JIC Module Added

SavedReadPlanConfig

- Support for M6e-JIC Module Added

SecureReadData

- Support for M6e-JIC Module Added

MercuryAPI 1.27.3

Support for USBPro Reader

The API now supports the USBPro reader. Aside from recognizing it as different from the Micro-LTE module alone, the API also works in concert with the reader’s hardware to support both GPI switch-sensing and GPO LED-illuminating functions. The internal
antenna is identified as “Antenna 1” and the external antenna is identified as “Antenna 2”.

**Added “Autonomous Mode” code sample**

An “Autonomous Mode” code sample has been added to illustrate how to create and store read plans on the module. Code samples already existed for saving configuration settings on the module.

**ATA Protocol**

ATA protocol support added to API. (Supported on Micro module with an optional license.) (Ref# 4744)
New Features of Universal Reader Assistant

The following section contains a brief summary of new features in Universal Reader Assistant. See the Universal Reader Assistant User Guide for detailed information.

Universal Reader Assistant 3.3

Ability to Install a Hex License Key
The ability to install license keys was added in version 3.2, but it could only be accomplished if the user had previously created a CSV-format license key file with the correct headers and format. Now, the license key panel will ask for the license string alone and this can be cut and pasted into the text-entry box.

Universal Reader Assistant 3.2

Dynamic Changing of Settings During Continuous Reading
Universal Reader Assistant can now demonstrate the ability to change settings during continuous reading. Any settings in the “Display Gen2 Settings” category can be altered, as well as the global read and write power levels (although write power is of limited use since the “write” tag operation cannot be specified under continuous reading in this version of Universal Reader Assistant).

Changes to the power levels are applied silently. Changes to Gen2 parameters result in a pop-up progress bar which disables further changes until the one you made is applied, as shown below.
New Features of Universal Reader Assistant

**Help and ToolTips Added**

A new “help” icon has been added to the top of the window, as shown here:

![Help Icon](image)

Protocol Result Column Displayed Automatically

When multiple protocols are selected, the ‘Protocol” results column automatically appears, as shown below.

Filtering on Tag Length and EPC Truncation Support

Universal Reader Assistant can now demonstrate filtering based on tag length and EPC truncation, as shown below.
Note that “EPC Truncate” is difficult to distinguish between a normal filter on EPC ID because the part of the EPC that is not reported by the tag is appended to the EPC as reported in the tag results screen.
ATA Protocol Support for Sargas

URA now supports Sargas readers which have been licensed to read the AEI ATA or IPX protocols.

The “Tag Inspector” tab has been revised so it can interpret the information in AEI ATA tags per the AAR S-918 encoding standard.
New Features of Universal Reader Assistant

License Installation Support [CHANGED IN VERSION 3.3]
The Firmware Update panel has been enhanced to be able to install license keys.

Serial Number Display
The serial number of the attached reader is now displayed in the Reader Diagnostics panel. This is the value you need to provide to ThingMagic if you are purchasing an optional license for your reader.
New Features of Universal Reader Assistant

Support for Local Readers
Universal Reader Assistant will now discover readers that are locally attached via Link Local addressing.

Connection Wizard Added
A Connection Wizard has been added to help users establish an initial connection to a device without the confusion of all the additional panels in the main Universal Reader Assistant screen. Users may still select “Cancel” when the Wizard comes up on Universal Reader Assistant start-up in order to access the main screen as before. Instructions for using the Wizard are in the Universal Reader Assistant User Guide.

Reader Diagnostics

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFID Engine</td>
<td>Sargas</td>
</tr>
<tr>
<td>COM/IP</td>
<td>192.168.7.2</td>
</tr>
<tr>
<td>Firmware Version</td>
<td>5.1.3.51</td>
</tr>
<tr>
<td>Hardware Version</td>
<td>1.0</td>
</tr>
<tr>
<td>Serial Number</td>
<td>50150123010004</td>
</tr>
<tr>
<td>URA Version</td>
<td>3.1.40.40</td>
</tr>
<tr>
<td>Mercury API Version</td>
<td>1.29.1.40</td>
</tr>
</tbody>
</table>
Automatic USB Driver Installation

As part of the Universal Reader Assistant installation process, signed USB drivers for the Micro (2-port) and M6e (4-port) modules are now installed, eliminating the need to download them from the ThingMagic web site and install them manually.

Custom Transport Support

Universal Reader Assistant now supports custom transport interfaces to serial readers. This capability is particularly useful when the connection to a serial device is made over a transparent network connection such as a TCP serial bridge. (If you select “Network Reader”, Universal Reader Assistant assumes you desire to communicate with the reader using the LLRP networked reader protocol.)
Expanded RF Power Settings

Support for both read and write power levels have been added to Universal Reader Assistant. In addition, the user can elect to set RF power levels per-port instead of universally.
New Features of Universal Reader Assistant

Temperature Updated More Regularly
The temperature is now updated more regularly. Here is a chart of when it is updated for various read operations:

<table>
<thead>
<tr>
<th>Read State</th>
<th>For Serial Module</th>
<th>For Network Reader</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial Connection to Reader</td>
<td>Shows current temperature</td>
<td></td>
</tr>
<tr>
<td>During Continuous Reading</td>
<td>Shows last measured value, doesn't update</td>
<td>No temperature shown</td>
</tr>
<tr>
<td>After Continuous Reading Is Complete</td>
<td>Shows temperature at the end of read cycle.</td>
<td></td>
</tr>
<tr>
<td>During Single, Timed Read</td>
<td>Shows last measured value, doesn't update</td>
<td>No temperature shown</td>
</tr>
<tr>
<td>After Single, Timed Read</td>
<td>Shows temperature at the end of read cycle.</td>
<td></td>
</tr>
</tbody>
</table>

Tag Aging Display is now Default
Previous versions of Universal Reader Assistant supported tag aging, where the tag results row will gradually darken as the time since the tag was last seen increases. This is now the default behavior (but still be turned off).

Enhancements to Error Logging
In previous versions of Universal Reader Assistant, error log files were kept for each session and each new connection. They were stored under “...\Documents\URA\” and had names that began with “tmrlog” followed by a date code. With the latest changes, there will be one error log file, shared by all instances of Universal Reader Assistant. It will be in the C:\URAlogs\” directory under the name “URA-Errorlogs.log”. The format of data entries will be:

[Timestamp] [LogType] : [ReaderUri] : [Message]
For example:

2016-07-19 11:50:11,473 [ERROR] : [COM35]: The module has exceeded the maximum or minimum operating temperature and will not allow an RF operation until it is back in range.

**Timestamp Handling**

Currently, all tag reads from serial readers and modules receive a system clock timestamp from the local PC, which is displayed by Universal Reader Assistant. Fixed readers have their own clocks, so the time for each tag entry is passed to Universal Reader Assistant, which then scales the time according to the following rules:

- If an M6 or Astra-EX reader provides the time in UTC units based on a central NTP server, Universal Reader Assistant converts the time zone to local time and displays it.
- If an M6 or Astra-EX reader has not synchronized with a central NTP server, Universal Reader Assistant will recognize this and display local PC time with the tag reads, much like it does for serial readers.
- For Sargas, the timestamp will always be considered to be in UTC time, which will be converted to the local timezone. If no NTP server is present to synchronize the time, the default time will start, at reboot, at a date in 2015.

**Reader Diagnostics Information Expanded**

The Reader Diagnostics page provides information about Universal Reader Assistant, the API on which it is based, and the reader to which it is connected. Previous releases gave the type of reader, firmware version of reader, hardware version of reader, URA version and API version, URA now also informs you of:

- COM port or IP address of the reader
- Serial number of the reader

**Universal Reader Assistant 2.8.16**

**“Regulatory Testing” panel added**

Three features were added with a new Regulatory Testing panel (to the right of the Tag Results display, below “Reader Diagnostics”.

- “CW” function that sends an un-modulated signal for the specified amount of time at the first frequency in the hop table.
New Features of Universal Reader Assistant

- “PRBS” which sends a randomly modulated signal for the specified amount of time at the first frequency in the hop table
- Hop Table. A comma-separated list of channel frequencies (with no spaces), in order of use. Each timed read cycle (or AsynchOnTime cycle) the reader will start at the first channel frequency and hop through all channels when desirable for best performance or mandated by regulatory requirements, returning to the first. The list may be truncated for test purposes or to limit channels to a range permitted by local regulatory requirements. Selecting a new region will reset the list to the default for that region.

Temperature Display Added
A temperature display has been added to the top of the screen. Its units are degrees C. If continuously reading (AsyncOffTime=0) the temperature will continually update each time the channel frequency changes. It will not update for a single timed read nor for continuous reading where AsyncOffTime is not zero.

USBPro Reader Support
Universal Reader Assistant detects when it is communicating with a USBPro reader as opposed to a Micro module and reacts accordingly, including:

- Added warning to set region if user tries to detect antennas using return loss method without selecting a region first (Ref# 4676)

Load/Save Improvements
The tag results column selection is no longer saved or loaded in keeping with its ability to be changed dynamically even when tags are being read (Ref # 4648)
New Feature of Autonomous Configuration Tool

Version 1.2.1 derived from API 1.29.3

(No change in features)

Version 1.2.1 derived from API 1.29.1.40

New Hardware Support
The Autonomous Configuration Tool now supports the M6e-JIC module

Duty Cycle Support
The Autonomous Configuration Tool now supports duty cycle control, to complement support in emerging versions of module firmware. This will allow the module firmware to control duty cycle to save battery life and reduce temperature rise.

<table>
<thead>
<tr>
<th>Embedded Read Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory Bank</td>
</tr>
<tr>
<td>On(ms) 1000</td>
</tr>
<tr>
<td>Duty Cycle</td>
</tr>
</tbody>
</table>

Note that the API has the ability to also set the hop table, hop time, “Q” value and TARI value for autonomous operation, but the Autonomous Configuration Tool does not support this yet.
Version 1.2.0 derived from API 1.27.3

Transport Logging Support
Option to enable Transport Logging has been added to the “Connect” panel.

New Feature of Autonomous Configuration Tool

More information provided in “About” screen
Additional information was added to the “About” panel (under the “Configure” tab):

- RFID Engine
- Firmware Version
- Hardware Version
- Autonomous Configuration Tool Version
- Mercury API Version

User Interface Enhancements
- A color-coded status indicator was added to the lower left of the display indicating “Not Connected” (red), “Connected” (orange) or “Reading” (green).
- The settings for reading when the unit is powered up vs. on GPI assertion were made clearer.
Bug Fixes

API 1.29.3

- Reading no longer ceases when a buffer-full exception is seen (Ref# 5243)
- Improved read consistency for async reading with SAMD21 “bare metal” processor (Ref# 5217)
- Changing duty-cycle and RF power level on-the-fly is now supported on “bare metal” platforms (Ref# 5195).

API 1.29.0

- "Error messages for IDS SL900A custom commands have been made more specific (Ref# 3536)
- "Error messages for invalid GPIO commands for Network readers made more specific (GPIO direction is fixed and cannot be changed as for modules.) (Ref# 4971)
- "Blockwrite code sample now supports antenna selection (Ref# 4801)
- "FastID Code sample now verifies that the mask length is correct before executing command. (Ref# 4799)
- ReaderStats code sample now enables antenna detection for the Micro so that the antenna connection status is reported (Ref# 5013)

Universal Reader Assistant 3.3

- Users can now copy information from the Reader Diagnostics panel (Ref# 5176)
- “Tag Inspector” can now indentify addional tag makes and models (Ref# 5080)

Universal Reader Assistant 3.2

- "URA will no longer default to reading Gen2 tags if no protocol is selected. (Ref# 4962)
- "Regulatory Testing controls now clearly indicate that they are not supported for fixed readers (Ref# 5002)
- "CW and PRBS transmission had been occurring on the first frequency of the default hop table for the region. Now they occur on the first frequency in the current hop table, even if it has been changed. (Ref# 4845)
• "URA correctly does not let the user access the "Untraceable" tab while reading is in progress. (Ref# 4890)

Universal Reader Assistant 2.8.16

• Individual tag operations were reporting “More than one tag responded” when continuous reading clearly showed only one tag in the field. This has been corrected. (Ref# 4593)
• User is now given the capability to upgraded firmware again if they inadvertently upgrade the firmware with a version that is not intended for that module (Ref# 4637)
• User is no longer being asked to re-select a region when moving from the continuous reading results page to the single tag operation tabs (Ref# 4743,4818)
• Corrected incorrect error being displayed when single tag operations were attempted without setting a region. (Ref# 4700)

Autonomous Configuration Tool version 1.2.1

• Better handling of situation where reader is already in autonomous mode when connection is attempted (Ref# 4751)
• Better handling of output power ranges that change with region (Ref# 4719)
• User now has the ability to turn off autonomous reading without having to restore all settings to default (Ref# 4713)
• Display screen when actively reading is now refreshed faster. (Ref# 4711)
• Restoring configuration files no longer results in misleading error messages (Ref# 4697)
• ThingMagic logo displayed more consistently (Ref# 4691)
• “About” and “Display Options” panes are no longer disabled after a firmware update (Ref# 4686)
• Loading a configuration that includes reading memory data no longer reverts to reading EPC data regardless of the saved configuration (Ref# 4685)
• Error message for configuration load errors is now more concise (Ref# 4683)
• Some re-connection issues following “Revert to Defaults” have been eliminated (Ref# 4660)
• Fixed issue where antenna detection was interfering with “Revert to Defaults” (Ref# 4657)
• Warning instead of error message now sent if Autonomous mode is enabled at the time the user tries to upgrade the firmware (Ref# 4653)
Outstanding Issues and Caveats

• Output power settings are no longer rounded down to the nearest dBm when saved - decimal values are now saved (Ref# 4633).
• In the case where a region value in a configuration file being loaded is not supported on the target module, the existing region will not be changed on the module (Ref# 4613)
• The saved default protocol is now being correctly loaded when a configuration is uploaded immediately after “Revert to Defaults” is applied (Ref# 4612).
• UI element for antenna selection is no longer grayed-out if a configuration file is loaded that has “checkPort=true” and “Antenna=1” (Ref# 4611)
• Non-supported Gen2 settings are no longer offered for the Nano (ref# 4608)
• Temperature is now displayed whether the reader is actively reading or not (Ref# 4603)
• Help topics were updated to include information for the Nano and USBPro. (They have not been updated to include the M6e module yet) (Ref# 4599)
• Choice of BLF-640 is added, with the “M” value correctly limited to “FM0” (Ref# 4573)
• Micro GPI choices are correctly limited to 2 pins (Ref# 4571)
• Antenna detection now works properly - if selected, it immediately checks to see which antennas are present and enables those that are (Ref# 4558)
• A success message is now displayed when “Revert to Defaults” is applied (Ref# 4518)
• Error reporting improved when invalid or blank configuration settings are loaded (Ref# 4275)

Outstanding Issues and Caveats

MercuryAPI v1.29.2

• The firmware released with this API does not tolerate extremely long Async RF Off times. The limit for the M6e and Micro is 11454. The limit for the Nano is 4295. (Ref# 5154, 5157, 5159)
• Be very careful when upgrading firmware that you are using the firmware that is appropriate for your reader or module. All firmware has a “SIM” file prefix, so the upgrade utility cannot check if the firmware is appropriate. Inappropriate firmware may cause the reader or module to stop responding to the MercuryAPI. In some cases, there are low-level tools that can be used to hold the module in its bootloader mode until the application firmware can be re-loaded. In other cases, the reader or module might not be recoverable in the field.
Bug Fixes

- Although antenna detection is now supported in the Micro module, it is not dynamic, and the code samples will continue to detect whether the module is a Micro and, if so, return an error message indicating that an antenna choice must be made.

- Java API creates a dependency on Log4j which is unnecessary. If you need to reduce the Java code size, replace this:

```java
private static Log4jLoggerAdapter logger;
logger = (Log4jLoggerAdapter)
LoggerFactory.getLogger(SerialReader.class)
```

With this:

```java
private static logger logger;
logger = LoggerFactory.getLogger(SerialReader.class)
```

(Ref# 4446)

- "C-API will throw an error, when user performs embedded read operation on M6, Micro or Sargas with filter when running Embeddedreadtid code sample. (Ref# 5016)

- "Java API will throw 'NullPointer Exception', when user try to read multiple memory banks data of a gen2 tag at a time when running the Gen2ReadAllMemoryBanks code sample. (Ref# 5014)

- There is no 'tari' validation in API for network readers. If BLF is set to “640 kHz”, then the Tari must be manually set to “6.25 usec” (the only Tari supported at that BLF. (Ref# 4874)

- "Java API will throw "'LTK XML message can not be validated against schema org/llrp/llrp-control-def.xsd', when we run RebootReader.java codelet on llrp readers by enabling transport listener. Happens when you run "RebootReader" codelet - only
works if transport listener is off. When error occurs, cannot reconnect to reader and read tags. (Ref# 3848)

**Universal Reader Assistant 3.2.1**

- URA does not display the option “Equal Time” as an antenna switching method if there is only one antenna on the reader. The USB Pro reader is erroneously included in this category, so the option is not available in this version for this reader. (Ref# 5160)
- URA won't show the network readers in reader's drop down list if your PC/Laptop is configured to WiFi. But, when user enters the IP address of the reader, a connection can be made. (Ref# 4963)
- Reading the entire contents of very large User memory tags (such at the Tego 24k tag) can lead to extremely large memory consumption by URA and occasional hangs. (Ref #4217)
- If URA is used to read a large tag population (greater than 150 tags) over a very long period of time, the URA refresh rate should be changed to 999 msec (set in Display Options panel). This gives the “garbage collection” routine enough time to prevent memory leakage. (Ref #3435)
- If a PC running URA is connected to the USB port of an M6e or Micro module and the power is removed while the module is continuously reading, the module will have to be rebooted before a connection can be reestablished. (Ref # 3581)
- If URA fails to run on a Windows 8 system, check the security levels of the file “C:\Windows\Microsoft.Net\Framework\V4.0.30319\config\machine.config”. It must have Read and Read&Execute permissions. If in Deny mode, URA will fail when launched. See [https://social.msdn.microsoft.com/Forums/vstudio/en-US/007f961c-9913-40ef-91d452632ccce3d/permissions-for-machineconfig-for-standard-user](https://social.msdn.microsoft.com/Forums/vstudio/en-US/007f961c-9913-40ef-91d452632ccce3d/permissions-for-machineconfig-for-standard-user). Here is another source of information: [https://learn.sparkfun.com/tutorials/disabling-driver-signature-on-windows-8/disabling-signed-driver-enforcement-on-windows-8](https://learn.sparkfun.com/tutorials/disabling-driver-signature-on-windows-8/disabling-signed-driver-enforcement-on-windows-8) (Ref #4408)

**Autonomous Configuration Tool version 1.2.1**

This version of the Autonomous Configuration Tool does not support the several of the new settings that can be saved in the latest versions of firmware. These settings that cannot be demonstrated using this tool are:

- hop table
- hop time
Outstanding Issues and Caveats

- "Q" value
- TARI value