

# RDS Translator

By Leif Claesson

Version 0.52

2018-03-13

RDS Translator is a simple but versatile system service to facilitate automatically updating the RDS and Streaming metadata parameters (such as the currently playing song title) in the Omnia.9, or any other product which supports metadata updates by HTTP push.

It can be run as a command line application (for testing) or as a system service.

**You will need to run it from a command prompt while setting it up.**

While running, you can see real-time status information the following ways:

- <http://127.0.0.1:7370>
- <telnet://127.0.0.1:32700>
- Console (command prompt test run only, system services do not have windows)

## What's New in v0.52

- HTTP push support (so you can push data to RdsTranslator which in turn pushes to anywhere). This makes it useful together with tools like **MagicRDS** which has a limitation that it can only push to one http URL (and it also cannot write to files). This way, RdsTranslator can push the data along to multiple Omnia.9's.
- Telnet console added for real-time status output.  
So, you can now get status output three ways: http (7370), telnet (32700), console (when testing from the command line only).
- http timeout ini-file option added – useful when you're pushing to more than one URL, so it won't wait too long for a non-responding server, delaying everything else.
- filter\_quotes option added, removes " quotation marks

## Preparation

The target system must have network access to the Omnia.9, and the IP of the target system must be in the Omnia.9's HTTP Access white list.

If you are able to visit **http://omnia9-ip:7380** in a web browser on the target system, then the target system has the required access.

The artist and song title must be available as a one file on the target system (or on a network drive). The file can be either XML or Plain Text.

## Installation

Copy RDS\_Translator.exe to a folder of your choice on the target system. We advise **not** putting it in inside Program Files to avoid UAC-related file redirection problems.

## Configuration

This is the tricky part, because it is up to *you* to define the configuration so that RDS translator can parse *your* data. This document has a complete description of the capabilities of RDS Translator, to allow you to create the configuration, but this document can not do your thinking for you. Getting two different systems to talk and interact is always tricky.

Without further ado, let's get down to business. RDS Translator will shortened to RTR in the rest of this document.

RTR will read its configuration from its own folder and file name, but with the ini extension.

Thus, if RTR is installed at C:\RTR\RDS\_Translator.exe, it will read C:\RTR\RDS\_Translator.ini.

RTR will continuously check the configuration file for updates, and will automatically re-initialize itself. You need never manually restart the service to apply a new configuration.

## XML Parsing

Let's say you're using AUTOMATON DJ STBH automation system, and it outputs an XML file with all pertinent info at c:\automaton\_dj\_stbh\now\_playing.xml

Here are example contents of the XML file:

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<PLAYBACKSTATE>
<TRIGGER>PLAY</TRIGGER>
<CURRENTTIME>12/07/12 12:34:56</CURRENTTIME>
<PLAYLIST>MUSIC</PLAYLIST>
<ENV />
<PLAY INDEX="0">
<CUTID>12346</CUTID>
<TITLE>Last night a DJ lost his job</TITLE>
<LENGTH>173.80</LENGTH>
<GROUP>A-ROTATION</GROUP>
<ARTIST>Indeed</ARTIST>
```

```
<ALBUM>Sign of the times</ALBUM>
</PLAY>
<REMAINING>00:03:20</REMAINING>
</PLAYBACKSTATE>
```

RTR can parse this quite easily.

Here is an example INI file:

```
[Input0]
file=c:\automaton_dj_stbh\now_playing.xml
type=xml
replace0=" ", " "
replace1="&", "and"
trim=1
uppercase=1
filter_accents=1

[Output0]
format0=Syndicate FM - %xml_play:title% - %xml_play:artist%
url0=http://127.0.0.1:7380/parameter/fm/rds/rds_rt=%format0%
```

## Input Section

We specify *type=xml* because the default is plain text, which uses entirely different processing.

You may use up to 100 replace statements to clean up the source data. In this example we replace double-space with single space, and & with and. Search-and-replace is case sensitive.

Many radios are extremely limited in what characters can be displayed. So, we use the following statements:

*filter\_accents=1* means for example *Mañana* becomes *Manana*.  
*uppercase=1* means *Manana* becomes *MANANA*  
*trim=1* means " MANANA " becomes "MANANA"

The processing order is always:

1. Retrieve Data
2. Filter Accents
3. Convert to Upper Case
4. Custom search-and-replace
5. Trim

The order of the statements in the [InputX] section does not affect the processing order.

This processing is applied individually to each retrieved XML reference in the *format* statement of the *output* section.

## Output Section

[Output0]

format0=Syndicate FM - %xml\_play:title% - %xml\_play:artist%

url0=http://127.0.0.1:7380/parameter/fm/rds/rds\_rt=%format0%

The syntax for the XML references is:

%xml\_key:key:key%

The key hierarchy can be any depth. Comparison starts at the deepest level.

Thus, in the following hierarchy:

```
<Playlist>Disco
  <Artist>Abba
    <Album>Abba Gold
      <SongTitle>Dancing Queen</SongTitle>
    </Album>
    <SongTitle>Knowing Me, Knowing You</SongTitle>
  </Artist>
</Playlist>
```

%xml\_artist:songtitle% fetches "Knowing Me, Knowing You"

But

%xml\_songtitle% and %xml\_album:songtitle% both fetch "Dancing Queen"

There is no need to write %xml\_playlist:artist:album:songtitle% even though you can.

A friend of mine once said: "You have a problem to solve. You bring XML in to solve it. Congratulations, now you have *two* problems."

## XML Properties

There are two distinct ways of storing XML data. Data can be plain text outside of tags, but it can also be properties *inside* tags.

Example:

```
<ParentTag>
  <Karaoke>
    <Song Artist="Abba" Title="Dancing Queen">
      Friday night and the lights are low
    </Song>
  </Karaoke>
</ParentTag>
```

In this case we can use `%xmlprop_hierarchy%` instead of `%xml_hierarchy%`.

`%xmlprop_parenttag:karaoke:song:Artist%` returns *Abba*

`%xmlprop_karaoke:song:Title%` returns "Dancing Queen"

You can mix `%xmlprop_%` and `%xml_%` at will to match your source data.

However, in the example above, due to the indentation, `%xml_karaoke:song%` returns  
[CR][TAB][TAB][TAB]Friday night and the lights are low[CR][TAB][TAB]

Computers are utterly unforgiving.

We can clean it up by adding the following statements to the Input section:

`replace0="[CR]", ""`

and

`replace1="[TAB]", ""`

It will then return just "Friday night and the lights are low".

## FORMAT specifiers

You can have up to 100 format specifiers, and you can enter any text you like around the XML tags.

For example, the following format specifier:

*format5=Now Playing at Syndicate FM: %xml\_artist% with %xml\_songtitle%*  
becomes

Now Playing at Syndicate FM: ABBA with DANCING QUEEN

The text in the format specifier is not processed by search/replace, trim, uppercase or any other input statements. What you write here is what you get.

If you wanted to change the format text at different times, for example having a different format for the morning show, you could set up a Windows Scheduled Task to call a command line utility like SetIni to update the format line in the RTR ini file. However, that's outside of the scope of this document.

## URLs

This is the final destination for the formatted data. You can have up to 100 URL specifications, and each one can reference as many format specifications as you want.

Example:

*url0=http://123.34.45.56:7380/parameter/fm/rds/rds\_rt=%format5%*

Any referenced format string will have special characters escaped.

The following format output:

*Now Playing at Syndicate FM: ABBA with DANCING QUEEN*

becomes

*Now%20Playing%20at%20Syndicate%20FM%3A%20ABBA%20with%20DANCING%20QUEEN*

So, the actual HTTP request that will be made is:

*url0=http://123.34.45.56:7380/parameter/fm/rds/rds\_rt=Now%20Playing%20at%20Syndicate%20FM%3A%20ABBA%20with%20DANCING%20QUEEN*

If you then look up the RDS RT parameter in the Omnia.9, you will see that it has been updated to "Now Playing at Syndicate FM: ABBA with DANCING QUEEN" which is exactly what we wanted.

## Plain Text Processing

Plain text parsing is tricky in a different way because there are no tags to go by. RTR offers several ways of getting the job done.

Here's the input data, at C:\AutomationSystem\NowPlaying.txt

```
Now Playing
Artist
"Ace of Base"
Song
All That She Wants
```

Example RTR configuration file:

```
[Input0]
file=c:\AutomationSystem\NowPlaying.txt
replace0=" ", " "
text0=find="Artist" line=1 uppercase=1 filter_accents=1
text1=find="Song" line=1 trim=1 filter_accents=1 filter_quotes=1
```

```
[Output0]
format0=%text0% - %text1%
url0=http://127.0.0.1:7380/parameter/fm/rds/rds_rt=%format0%
```

```
text0 returns "ACE OF BASE"
text1 returns "All That She Wants"
```

Thus format0 returns "ACE OF BASE - All That She Wants".

*find* searches for the line that contains the specified text.  
*line* then adds an offset. In this case, we're using that to get the *next* line, after the one that contained "Artist".

*find* is of course optional.  
*text0=line=0* would return "Now Playing"  
*text0=line=2* would return "Ace of Base"

You can also search within a line.

Example data:

Now Playing

Artist: Ace of Base Song: All That She Wants

*text0=*line=1 *begin="Artist:" end="Song:" trim=1* returns "Ace of Base"

*text1=*line=1 *begin="Song:" trim=1* returns "All That She Wants"

Thus, *format=%text0% - %text1%* returns "Ace of Base - All That She Wants"

*trim=1* removes spaces before and after the resulting text.

You may also use *uppercase=1* and *filter\_accents=1* which function just as with XML data.

FIXME: Add description for *rbegin* and *rend*

HTTP PUSH support

Allows RDS translator to accept data through HTTP GET queries.

Works similar to Plain Text processing, except the type is push, and the line number is always 0.

You can then push data to RDS translator as follows:

<http://127.0.0.1:7370/input0?set=whatever+data%20you%20like>

Example:

**type=push**

**Text0=line=0**



# Application-specific Recommendations

## RDS PS (Programme Service)

Most RDS-radios display only the 8-character PS (Programme Service) field. This field was never designed to be dynamic. However, people use it that way anyway.

The Omnia.9 has a few features to facilitate this.

For example, the following RDS PS string in the Omnia.9:

```
"<<   Daddy Yankee - Gasolina   /10s:FUTURAFM/2s:CALL US/4s:555-0911"
```

Will result in:

"Daddy Yankee - Gasolina" scrolls across the display 2 characters at a time.

"FUTURAFM" is displayed for 10 seconds

"CALL US" is displayed for 2 seconds

"555-0911" is displayed for 4 seconds

Notice the extra spaces around the title in the PS string -- this is necessary to make the text enter from outside the display. An example format string for this might be:

```
format0="<<   %xml_artist% - %xml_title%   /10s:FUTURAFM/2s:CALL US" etc.
```

## **RDS RT (RadioText)**

RadioText doesn't need nearly as much trickery. It can send 64 characters for a radio to display in any way it likes. Some radios scroll smoothly, others have a big enough screen to display the whole message, and some display when the listener presses the RT button.

## **Streaming Metadata**

Streaming metadata should not include the station name at all, because it is already included in the stream header. Best practice is to simply include the artist and song title and leave it at that. Web address (URL), Station Name, Genre all have their own dedicated fields and do not need to be updated in real time.

Because RTR supports multiple Input and Output sections, you can read multiple files, or even the same file multiple times, from a single RTR service.

Input0 belongs to Output0, Input1 belongs to Output1 etc.  
Output1 can only read data from Input1.

## Reading files from a network drive

RTR will happily read a file from a network drive, but when running as a service, special care must be taken.

Drive letter mappings (map network drive) will not be accessible from system services. You must:

- Specify the file name as [\\server\share\path\filename.txt](#)
- Install the service by executing RdsTranslator.exe -i
- Make the RDS translator service log in as your regular user name:
  - Open the service control manager (services.msc)
  - Find the RDS translator service in the list
  - Right-click, select Properties
  - Click on the Log On tab
  - Log on as: This Account
  - Browse
    - Enter your user name
    - Click Check Names
    - Click OK
  - Enter your password
  - Confirm your password
  - OK
  - Right-click the service in the list, select Restart

## Testing and Finishing

RTR can run as either a command line application or as a service.

Type *RDS\_Translator ?* for a list of command line options.

We recommend running as a command line app when testing, because you then have immediate feedback in the console.

However, when you have the configuration set up the way you want to, it's much better to run RTR as a service, because then there is no window to get in the way, and you do not even have to be logged into the computer.

Even when running as a service, you can get runtime info by visiting the built in HTTP server at *http://RTR-computer-ip:7370* from a web browser. Remember to add a firewall exception for RTR if you want to access it over the network. Locally, you can access *http://127.0.0.1:7370* and no firewall exception is necessary.

# Configuration File Command Reference

## Plain Text

```
[InputX]
file=input filename
type=text
replaceX="find", "replace"
textX=find="Artist" line=0 uppercase=1 filter_accents=1 trim=1 begin="" end=""
textX=find="Artist" line=0 uppercase=1 filter_accents=1 trim=1 rbegin="" rend=""
```

```
[OutputX]
formatX=%textX%
urlX=http://url/url=%formatX%
```

## XML

```
[InputX]
file=input filename
type=xml
replaceX="find", "replace"
uppercase=1
filter_accents=1
trim=1
```

```
[OutputX]
formatX=%xml_parent:child:grandchild% - %xmlprop_parent:child:property%
urlX=http://127.0.0.1:7380/parameter/fm/rds/rds_ps=%formatX%
```

## HTTP push

```
[InputX]
type=push
replaceX="find", "replace"
textX=find="Artist" line=0 uppercase=1 filter_accents=1 trim=1 begin="" end=""
textX=find="Artist" line=0 uppercase=1 filter_accents=1 trim=1 rbegin="" rend=""
```

```
[OutputX]
formatX=%textX%
urlX=http://url/url=%formatX%
```