RTSS User's Manual

For RTSS Version 2.6.4

DR RYO TAKAGI

2011. 1. 12

Table of Contents

XML Data File Lists

1 Overview

1.1 What is RTSS?

RTSS is a flexible tool for the simulation and evaluation of the electric power feeding network for DC electric railways. The name RTSS is short for Railway Total System Simulator, intended to be a more comprehensive simulation tool of the whole railway system.

The name RTSS is also known as the abbreviation of the authors' names --- Dr Ryo Takagi, who is now with Kogakuin University in Tokyo, Japan as Associate Professor, and Professor Satoru Sone, Professor Emeritus of the University of Tokyo and a figure well-known throughout the world in the field of electric railway engineering.

1.2 What is the history of RTSS?

RTSS has been developed as part of the PhD project of Dr Ryo Takagi. The PhD thesis was (To be completed.)

1.3 What is the track record of RTSS?

(To be completed.)

1.4 Who should read this manual?

(To be completed.)

2 Steps to Use RTSS

2.1 Preparing the computer environment

(To be completed.)

2.2 Finding the distribution site

You can find the newest distribution of RTSS at the following URI:

`http://www.takagi-ryo.ac/p/railway/rtss`

Here you can find the source and binary distributions of every RTSS release, information on how RTSS has changed since the last release, manuals and other useful information.

2.3 Compiling and installing the program

RTSS is written in standard C++. It will probably run on almost all platforms provided an ISO standard compliant C++ compiler is available thereon. There is also a binary distribution for the Microsoft Windows environment. For more information see Chapter 3 of this manual.

2.4 Preparing data files

RTSS requires a considerable amount of information to run a simulation. Such information can be given through input XML files. For the grammar of the input files and how you can use them, see Chapter 4 of this manual.

2.5 Invoking the program

(To be completed.)

2.6 Looking into the simulation results

(To be completed.)

3 Compiling and Installing RTSS

3.1 Compiling from source tarball

If you want to compile RTSS from the source tarball (source distribution), you will need a C++ compiler and the environment that supports it. You will also need the PCRE library, which should be precompiled, installed and ready to use.

The author is using the latest MinGW / MSYS package installed on a personal computer running Microsoft Windows Vista (both 32-bit and 64-bit versions).

(1) Unpack the source tarball. Using the "tar" command, All files in the distribution will be unpacked into a single directory called "`RTSS-x.y.z`", where `x.y.z` is the version number of RTSS.

(2) Create a directory called "`lib`" under the home directory if it is nonexistent.

(3) In the source distribution, locate the makefile to be used during compilation. There are several makefiles in the source distribution. `Makefile` and `Makefile-MinGW` are the choices; `XMakefile` will be called during compilation using either of these makefiles. If you are using MinGW / MSYS, `Makefile-MinGW` is the one for you. If you are using Unix or Linux systems, use `Makefile` instead.

(4) Open the appropriate makefile using your favourite editor, look through it and make necessary changes. If you are to use `Makefile-MinGW`, change the line defining `HOME` macro to the correct home directory of yours. When you have done the edits, save the makefile.

(5) Go down to the directory where you have unpacked the tarball (RTSS-x.y.z), and type "`make –f NAME_OF_YOUR_MAKEFILE`". Here, `NAME_OF_YOUR_MAKEFILE` depends on your choice of makefile in step (3).

(6) RTSS binary executable file will be generated in the directory called "`obj-RTSS-x.y.z`", which is located in the same directory as the one RTSS-x.y.z resides. You can move RTSS executable file to wherever you want, provided the PCRE shared library resides in the place that the executable can find. If you are using Microsoft Windows, you may want to copy "libpcre-0.dll" together with the executable.

3.2 Installing using the Windows installer

In the RTSS distribution website, an installer for the Microsoft Windows environment (probably usable only in Windows XP and newer versions) is provided. Just download the file and execute it; the result is that the binary executable file, together with the necessary libpcre-0.dll, will be installed to wherever directory you want ("`C:¥rtss`" by default).

The binary executable file that will be installed by this installer is compiled using the 32-bit version of MinGW.

You may have trouble downloading the installer, since some antivirus or firewall software rejects the file. In this case you will have to disable the software temporarily before performing the download.

3.3 Finding the version of the RTSS executable

You may want to know the version number of the executable that you have found on your system. If the binary is of Version 2.6.2 or newer, you can find it by giving RTSS executable the "`-o -v`" command line options.

4 Preparing Data Files

Before you run RTSS, you must be familiar with the simulation model implemented in it. This is not an "ordinary" type of software --- you must have the entire knowledge on what you are doing! The information you need is given in the PhD thesis (1995) by the author, which can be downloaded from the RTSS website.

4.1 General information on the XML format

All data files will have the XML format. A data file will look like List 4:1.

List 4:1 An example XML file

```
001: <?xml version="1.0" encoding="UTF-8"?>
002: <!-- Comment -->
003: <rtss_A_tag>
004:   <rtss_B_tag>
005:     <rtss_C_tag />
006:   </rtss_B_tag>
007:   <rtss_D_tag name="D" distance="1.0" />
008:   <rtss_E_tag name="E">
009:     <rtss_F_tag />
010:   </rtss_E_tag>
011:   <rtss_G_tag name="G">
012:     <rtss_H_tag distance="1.0" />
013:     <rtss_H_tag distance="2.0" />
014:   </rtss_G_tag>
015: </rtss_A_tag>
```

Here, line 1 is the XML declaration. An XML file must have this at the beginning of it.

In the lines that follow this XML declaration line, the strings surrounded by angle brackets are called tags. On line 3 of List 4:1, you can find tag `<rtss_A_tag>`, which is called the start tag. This corresponds to the end tag, which is `</rtss_A_tag>` on line 8. Note that end tags start with beginning angle bracket followed by a slash. The combination of a start tag and an end tag is called an element.

Between the start tag and the end tag, there may be contents of the element. The

contents can either be any text strings, comments, and child elements, or the combination of these. For example, <rtss_B_tag> on line 4 of List 4:1 is another start tag, which corresponds to the end tag on line 6. This element is the child element of <rtss_A_tag>.

If there are no contents in the element, you can simplify the expression by using the empty-element tag instead of a pair of start and end tags. An empty-element tag can be found in line 5 of List 4:1. Note the slash before the ending angle bracket in line 5. This expression is equal to <rtss_C_tag></rtss_C_tag>.

An element can have one or more attributes. An empty-element tag with two attributes can be found on line 7 of List 4:1. Here, element <rtss_D_tag> has two attributes, name and distance. As shown here, each attribute will be given a value, which is a text string surrounded by single or double quotes. Similarly, a start tag with an attribute can be found in line 8, where element <rtss_E_tag> has attribute name. Note that element <rtss_E_tag> also has a child element, namely <rtss_F_tag>. End tags must not have attributes. Attributes with the same name must not coincide within one tag; for example, the expression below will cause an error:

```
<rtss_G_tag name="G" distance="1.0" distance="2.0" />
```

One way of specifying values named "distance" twice (or more) is to have two (or more) child elements, as in lines 11 through 14 of List 4:1. Here, instead of having two distance attributes in the tag, element <rtss_G_tag> has two <rtss_H_tag> child elements, each with an attribute distance.

Also note that the order in which attributes appear within a tag is not important; you may put them in any order you want. On the contrary, the order in which elements appear within the XML file may mean something.

All XML documents must have one "root element". This is an element in an XML document which is not a child of any other elements in the document. In List 4:1, <rtss_A_tag> is the root element.

On line 2 of List 4:1 is a comment. Comments are surrounded by <!-- and -->. Inside these you can put any kind of text strings except "--" (double minus). You can put any number of comments as contents of an element, or outside the root element of the document.

The encoding of the entire XML file must be "UTF-8" (Unicode).

In the data files of RTSS, any data will be given as attributes of a specific element, and any one element will either contain child element(s) and comments, or be empty.

4.2 Types of input and output data files

To run RTSS, you must prepare the following input data files:

(1) **Control File**. This file will specify the paths of all other input and output data files.

(2) **G File**. This file will specify track profile information (gradient, curvature and permanent speed restriction).

(3) **N File**. This file will specify train schedule data.

(4) **P File**. This file will specify train performance data.

(5) **S File**. This file will specify substation characteristics data.

(6) **F File**. This file will specify the feeding network information and overall simulation parameters and options.

There are three output files that will be created after you run RTSS:

(7) **R File**. This file is a simple text file with the summary of simulation results.

(8) **C File**. This file is a "CSV" (comma-separated variables) file with all detailed simulation results, including records of simulated position of trains, voltage and current of substations and trains, etc. It is a very large file. Microsoft Excel is designed to handle CSV files, but sometimes the C File can grow too large for older version of Excel to handle.

(9) **L File**. This file is a simple text file containing log messages. It is a very large file.

4.3 Control File

This file will specify the paths to all the input / output files. An example of the Control File can be found in List 4:2.

List 4:2 Control File

```
001: <?xml version="1.0" encoding="UTF-8"?>
002: <rtss_controls>
003:   <rtssfile type="F" name="../f/Ffile.xml" />
004:   <rtssfile type="G" name="../g/Gfile.xml" />
005:   <rtssfile type="N" name="../n/Nfile.xml" />
006:   <rtssfile type="P" name="../p/Pfile.xml" />
007:   <rtssfile type="S" name="../s/Sfile.xml" />
008:   <rtssfile type="R" name="../res/res.rtss" />
009:   <rtssfile type="C" name="../csv/res.csv" />
010:   <rtssfile type="L" name="../log/res.log" />
011: </rtss_controls>
```

4.3.1 <rtss_controls>

This element is the root element of the Control File.

This element can contain one or more <rtssfile> elements. It must not have any attributes. It is recommended that this element contains eight <rtssfile> elements, containing the paths to all the input / output files.

4.3.2 <rtssfile>

This element is the child of <rtss_controls> element.

This element must be empty. It must have two attributes, type and name.

(1) type: One of "G", "N", "S", "P", "F", "R", "C", "L", each corresponding to one of eight input and output files, G File, N File, etc., is to be specified as the value of this attribute. Any other values will cause an error. This value will show that this element specifies file path of the particular input/output file.

(2) name: The file path to the File specified by type attribute will be given as value of this attribute. The path can be relative to the path of Control File itself.

There can be up to eight appearances of <rtssfile> elements within the <rtss_control> element, each with different value of type attribute. If there are two or more appearances of <rtssfile> elements with the same type attribute value, the result is undefined.

Note that all input files, i.e. files with types "G", "N", "S", "P" and "F", must have names that end with ".xml".

Also note that types "R", "C" and "L" are output files. The directory (or directories) that will accommodate these files must be created before running RTSS; it will not be created automatically by RTSS itself.

4.4 G File

This file will specify the track profile information, including gradient, curvature and speed restrictions of the railway line to be simulated. An example of the G File can be found in List 4:3.

List 4:3 G File

```
001: <?xml version="1.0" encoding="UTF-8"?>
002: <rtss_g_file>
003:   <speed_restriction_margin value="0.0" />
004:   <gradcrv_data_set name="G_01">
005:     <gradient>
006:       <data from="0.0" to="0.5" value="0.0" />
007:       <data from="0.5" to="1.0" value="10.0" />
008:     </gradient>
009:     <curvature>
010:       <data from="0.0" to="1.0" value="0.0" />
011:     </curvature>
012:     <speed_restriction>
013:       <data from="0.0" to="1.0" value="0.0" />
014:     </speed_restriction>
015:   </gradcrv_data_set>
016:   <gradcrv_data_set name="G_02">
017:     <three_data>
018:       <data from="1.0" to="2.0" gradient="-10.0"
019:             curvature="800.0" speed_restriction="70.0" />
020:     </three_data>
021:   </gradcrv_data_set>
022: </rtss_g_file>
```

4.4.1 <rtss_g_file>

This element is the root element of the G File.

This element can contain the following elements in the order shown below:

I. Up to one <speed_restriction_margin> element; and

II. One or more <gradcrv_data_set> elements.

This element must not have any attributes.

4.4.2 <speed_restriction_margin>

This element is the child of <rtss_g_file> element. This element must be empty. This element must have one attribute called value.

(1) value: A double precision figure given here as value to this attribute will be subtracted from all speed restriction data given in G File.

This element is optional and can be omitted. It must not appear after any one

<gradcrv_data_set> element appears in the <rtss_g_file> element. If two or more of <speed_restriction_margin> elements appear within the <rtss_g_file> element, an error occurs and the program will terminate.

4.4.3 <gradcrv_data_set>

This element is the child of <rtss_g_file> element. There can be any number of <gradcrv_data_set> elements within the <rtss_g_file> element. This element will define one gradcrv data set, a set of track profile information which includes gradient, curvature and speed restriction data.

This element must have one attribute called name.

(1) name: This attribute is given the name in string of this particular gradcrv data set.

If there are more than one <gradcrv_data_set> elements within the <rtss_g_file> element, every <gradcrv_data_set> element must be given a unique name.

This element can contain one of the following sets of elements shown below:

I. One <gradient> element, one <curvature> element and one <speed_restriction> element; or

II. One <three_data> element.

If I. of the above sets is selected, the three elements must cover the same section of track; this means that (1) the position specified by the attribute from of the first <data> child elements within the three elements, <gradient>, <curvature> and <speed_restriction>, are equal, and similarly (2) the position specified by the attribute to of the last <data> child elements within the three elements are equal. The three elements, <gradient>, <curvature> and <speed_restriction>, can appear in any order.

4.4.4 <gradient>

This element is the child of <gradcrv_data_set> element. There can be only one <gradient> element within the <gradcrv_data_set> element. If this element is present, then there must be no <three_data> element within the same <gradcrv_data_set> element.

This element can have one attribute called ref. When this attribute is specified, the element must be empty.

(1) ref: This attribute is given the name in string of another <gradcrv_data_set> element, from which the gradient data is copied.

If there is no attributes, then there must be one or more <data> child elements within this element. The <data> child elements must be continuous, i.e. the position

specified by the attribute to of a <data> child element must be equal to the position specified by the attribute from of the <data> child element which comes next to the one in question.

4.4.5 <curvature>

This element is the child of <gradcrv_data_set> element. There can be only one <curvature> element within the <gradcrv_data_set> element. If this element is present, then there must be no <three_data> element within the same <gradcrv_data_set> element.

This element can have one attribute called ref. When this attribute is specified, the element must be empty.

(1) ref: This attribute is given the name in string of another <gradcrv_data_set> element, from which the curvature data is copied.

If there is no attributes, then there must be one or more <data> child elements within this element. The <data> child elements must be continuous, i.e. the position specified by the attribute to of a <data> child element must be equal to the position specified by the attribute from of the <data> child element which comes next to the one in question.

4.4.6 <speed_restriction>

This element is the child of <gradcrv_data_set> element. There can be only one <speed_restriction> element within the <gradcrv_data_set> element. If this element is present, then there must be no <three_data> element within the same <gradcrv_data_set> element.

This element can have one attribute called ref. When this attribute is specified, the element must be empty.

(1) ref: This attribute is given the name in string of another <gradcrv_data_set> element, from which the speed restriction data is copied.

If there is no attributes, then there must be one or more <data> child elements within this element. The <data> child elements must be continuous, i.e. the position specified by the attribute to of a <data> child element must be equal to the position specified by the attribute from of the <data> child element which comes next to the one in question.

4.4.7 <three_data>

This element is the child of <gradcrv_data_set> element. There can be only one <three_data> element within the <gradcrv_data_set> element. If this element is present, then there must be no <gradient>, <curvature> or <speed_restriction> elements within the same <gradcrv_data_set> element.

This element must not have any attributes. There must be one or more <data> child elements within this element. The <data> child elements must be continuous, i.e. the position specified by the attribute to of a <data> child element must be equal to the position specified by the attribute from of the <data> child element which comes next to the one in question.

4.4.8 <data>

This element is the child of either <gradient>, <curvature>, <speed_restriction> or <three_data> elements.

This element must be empty.

If this element is the child of <gradient>, <curvature> or <speed_restriction> elements, then there must be **three** attributes called from, to and value. The use of attributes gradient, curvature and speed_restriction in this case is prohibited and will cause an error.

If this element is the child of <three_data> element, then there must be **five** attributes called from, to, gradient, curvature and speed_restriction. The use of attribute value in this case is prohibited and will cause an error.

(1) from: This attribute is given a double precision value in [km], which is the starting position of this data.

(2) to: This attribute is given a double precision value in [km], which is the ending position of this data.

(3) value: This attribute is given a double precision value.
If this element is the child of <gradient> element, it represents the track gradient (in [‰] = per mill, 1 ‰ = 1 in 1000) between positions specified by attributes from and to.
If this element is the child of <curvature> element, it represents the curve radius (in [m]) between positions specified by attributes from and to (if the value is zero then it will mean the track is straight).
If this element is the child of <speed_restriction> element, it represents the speed restriction (in [km / h]) between positions specified by attributes from and to (if the value is zero then it will mean there is no speed restriction).

(4) gradient: This attribute is given a double precision value, representing the track gradient (in [‰] = per mill, 1 ‰ = 1 in 1000) between positions specified by attributes from and to.

(5) curvature: This attribute is given a double precision value, representing the curve radius (in [m]) between positions specified by attributes from and to (if the value is zero then it will mean the track is straight).

(6) `speed_restriction`: This attribute is given a double precision value, representing the speed restriction (in [km / h]) between positions specified by attributes `from` and `to` (if the value is zero then it will mean there is no speed restriction).

4.5 N File

This file will specify the train schedule, or more informally, "how trains run". An example of the N File can be found in List 4:3.

List 4:4 N File

```
001: <?xml version="1.0" encoding="UTF-8"?>
002: <rtss_n_file>
003:   <n_flags ontprec="true" />
004:   <accompanying_file name="Ref_SOC_curve.xml" />
005:   <pattern cycletime="200.0" phase="0.0" cars="1"
006:           train_class="Class_465" patterncirc="1"
007:           init_train_position="0.0" init_train_velocity="0.0"
008:           init_train_state="Power" init_train_delay="0.0">
009:     <nextsta start_point="0.0" end_point="1.5" end_velocity="0.0"
010:             start_stop_time="150.0" dwell_time="50.0"
011:             notch_off_velocity="0.0" direction="1"
012:             feeder_name="Line_A_Up">
013:       <gradcrv name="G_01" />
014:     </nextsta>
015:   </pattern>
016: </rtss_n_file>
```

4.5.1 `<rtss_n_file>`

This element is the root element of the N File.

This element can contain the following elements in the order shown below:

I. Up to one `<n_flags>` element;
II. Any number (zero or more) of `<accompanying_file>` elements; and
III. One or more `<pattern>` elements (at least one `<pattern>` element is required).

This element must not have any attributes.

4.5.2 `<n_flags>`

This element is the child element of the `<rtss_n_file>` element. This element is optional and can be omitted. If present, it must be empty.

If this element is to appear, it must be put as the first element within the `<rtss_n_file>` element.

This element can have the following attributes. All attributes are optional and can be omitted.

(1) `conges_station`: This attribute is given a boolean value ("true" or "false"). If true, congestion can be specified individually for each `nextsta` data set.

(2) `ontprec_delay_set`: This attribute is given a double precision value in [s]. If the delay of a train is found to have exceeded this value during simulation despite the use of "precise start-to-stop time simulation" algorithm, the number of free-run cycles will be incremented by one, and the use of approximate start-to-stop time estimation routine will be suppressed in the subsequent simulation. The default value of `ontprec_delay_set` is 0.6 seconds.

(3) `ontprec`: This attribute is given a boolean value ("true" or "false"). If true, the use of approximate start-to-stop time estimation routine will be suppressed. This will result in longer calculation time. If the number of free-run cycles is set to zero, it is recommended that this attribute be given the value "true".

(4) `initialdeparturetime`: This attribute is given a boolean value ("true" or "false"). If true, all the `<pattern>` elements within this N File must have attribute `initial_train_departure_time`. This is used to "tweak" the schedule data; more details can be found in the author's PhD thesis.

(5) `startstop_parameter`: This attribute is given a double precision value in [s]. The value will be subtracted from the start-to-stop time specified in all `nextsta` data sets in this N File. At the same time, the value will be added to the dwell time specified in all `nextsta` data sets in this N File, so the cycle time of any one diagram pattern in this N File remains the same. The default value of this parameter is zero.

(6) `station_object_valid`: This attribute is given a boolean value ("true" or "false"). If set to true, the "station object" will be valid. This part of RTSS is not actively maintained or supported; this attribute must not be set to true unless you read the source code and have enough knowledge on the algorithm.

4.5.3 `<accompanying_file>`

This element is the child element of the `<rtss_n_file>` element. This element is optional and can be omitted. If present, it must be empty.

If this element is to appear within the `<rtss_n_file>` element, it must be put before the appearance of any one `<pattern>` element. Any number of `<accompanying_file>` elements can appear within `<rtss_n_file>`.

This element must have an attribute called `name`.

(1) `name`: This attribute is given a name in string of the path to the "accompanying

file" to be read by RTSS. At present, this file is used to describe data of the reference SOC curves, which are used in the control of on-board energy storage systems.

4.5.4 <pattern>

This element is the child element of the <rtss_n_file> element. Any number of <pattern> elements can appear within <rtss_n_file>; however, at least one appearance is required.

A <pattern> element will represent a diagram pattern, which is actually the queue of nextsta data sets. A nextsta data set contains the data required to run a train between two stations, and therefore a diagram pattern is a set of data that determines how train(s) run in the railway network to be simulated. In the simulation, every train will belong to one of the diagram patterns. In the diagram pattern, trains run cyclically; when a train finishes a run between two stations defined by a nextsta data set in the queue in the diagram pattern, the next nextsta data set in the queue will be used to run the train. When the train finishes the run defined by the last nextsta data set in the queue, it will move back to the first data set and continue. If there are two or more trains that belong to one diagram pattern, they are initially positioned before the simulation starts so that the time distance between any two adjacent trains is uniform.

This element must contain the following elements shown below:

I. One or more <nextsta> elements.

This element can have the attributes listed hereunder. Among these, attributes (1) through (9) are required and cannot be omitted. If attribute initialdeparturetime is set to "true" in the <n_flags> element, then attribute (10) is also required.

(1) cycletime: This attribute is given a double precision value, which represents the total time in [s] (seconds) for a train to complete one cycle of journey defined by this element. This is called the cycle time of this diagram pattern. This is in fact a redundant information, because it can be calculated by adding up the running times and the dwell times specified in all nextsta data sets in the queue; however, the cycle time information is not easy to calculate by hand, and therefore this attribute must be specified for double-checking the validity of the data.

(2) phase: This attribute is given a double precision value, which represents the phase in [s] (seconds) of this diagram pattern. The phase of the diagram pattern is the departure time of a train from the departing station specified by the first nextsta data set in the queue. It is recommended that the phase is zero when

there is only one diagram pattern in N File; the non-zero phase makes sense only when there are two or more diagram patterns.

(3) `cars`: This attribute is given an integer value, which represents the number of trains that belong to this diagram pattern.

(4) `train_class`: This attribute is given a string value, which represents the name of the train type defined in P File. All trains that belong to this diagram pattern will be of this train type.

(5) `patterncirc`: This attribute is given an integer value, which is called the pattern multiplier. It is given so that the following value T_S is the same for all diagram patterns in N File:

$$T_S = \frac{T_C}{N_T} \times M_P$$

Where T_C is the cycle time specified by attribute `cycletime`, N_T is the number of trains specified by attribute `cars`, and M_P is the pattern multiplier specified by this attribute.

It is recommended that the pattern multiplier is set to 1 when there is only one diagram pattern in N File.

The time T_S calculated by the above equation will be the cycle of simulation; when T_S seconds have passed in the simulation, it can be assumed that the condition of the system being simulated roughly returns to where it has started.

(6) `init_train_position`: This attribute is given a double precision value, which represents the initial train position in [km].

(7) `init_train_velocity`: This attribute is given a double precision value, which represents the initial train velocity in [km/h].

(8) `init_train_state`: This attribute is given a string or integer value, which represents the initial train state. The train state must be one of the following:

- `Power` or `0`: Powering (accelerating);
- `Brake` or `1`: Braking;
- `Coast` or `2`: Coasting;
- `Stn_stop` or `3`: Dwelling at station;
- `Const_vel` or `4`: Running at constant velocity; or
- `Regenerate` or `5`: Decelerating using regenerative brake only.

(9) `init_train_delay`: This attribute is given a double precision value, which represents the initial delay of the train in [s].

(10) `init_train_departure_time`: This attribute is given a double precision value, which represents the initial train departure time in [s]. Note that the

non-zero value given to this attribute makes sense only when the initial state of the train is set to Stn_stop (dwelling at station). This data can be set to whatever value, regardless of the dwell time specified by the nextsta data sets.

4.5.5 <nextsta>

This element is the child element of the <pattern> element. Any number of <nextsta> elements can appear within <pattern>; however, at least one appearance is required.

A <nextsta> element will represent a nextsta data set. A nextsta data set contains the data required to run a train between two stations.

This element must contain the following elements shown below:

I. One or more <gradcrv> elements.

This element can have the attributes listed hereunder. Among these, attributes (1) through (9) are required and cannot be omitted. If attribute conges_station is set to "true" in the <n_flags> element, then attribute (10) is also required. If attribute station_object_valid is set to "true" in the <n_flags> element, then attribute (11) is also required. Attributes (12) and (13) are optional and can be omitted.

(1) start_point: This attribute is given a double precision value, which represents the position in [km] at which this nextsta data set will start. Generally, this point is the position of the origin (starting) station.

(2) end_point: This attribute is given a double precision value, which represents the position in [km] at which this nextsta data set will end. Generally, this point is the position of the destination station.

(3) end_velocity: This attribute is given a double precision value, which represents the velocity in [km/h] at the end position. If this train is to call at the destination station, this must be zero.

This attribute must be given a non-zero value only when either Station_pass or Pass_pass is specified as the value of attribute nextsta_pattern in this element. When this is the case, the train will apply brakes toward the last part of the inter-station journey defined by this nextsta data set so that the train velocity does not exceed the value specified by this attribute at end position; if the velocity at end position will not exceed the value specified by this attribute, the train will simply coast without accelerating.

(4) start_stop_time: This attribute is given a double precision value, which represents the time in [s] (seconds) required for the inter-station run defined by this nextsta data set. If Station is specified as the value of attribute nextsta_pattern in this element, then it is the start-to-stop time between the

starting point and the end point. Otherwise, it represents the time required for this train to run between the starting point and the end point.

(5) `dwell_time`: This attribute is given a double precision value, which represents the dwell time in [s] (seconds) at the destination station. If either `Station_pass` or `Pass_pass` is specified as the value of attribute `nextsta_pattern` in this element, the value must be zero.

(6) `notch_off_velocity`: This attribute must be given value "0.0". This is the dummy parameter that has no effect, but cannot be omitted.

(7) `direction`: This attribute shows the direction of the train, and is given either `1` or `-1`. If "1" is given, then the train will run in "positive" direction, i.e. the position will increase when the train moves forward. If "−1" (minus one) is given, then the train will run in "negative" direction, i.e. the position will decrease when the train moves forward.

(8) `feeder_name`: This attribute is given a string value, which represents the name of the `feedline` object that this train will run under. The named `feedline` object must be defined in F File; otherwise it will cause an error.

(9) `nextsta_pattern` : This attribute is given a string value, representing the nature of this `nextsta` data set. Its value must be chosen from the list below:

- `Station`: In this data set, the train will start from the origin station (from zero velocity) and arrive (call) at the destination station;
- `Station_pass`: In this data set, the train will start from the origin station (from zero velocity) and pass the destination station without calling at it;
- `Pass_station`: In this data set, the train will pass through the origin station without calling at it and arrive (call) at the destination station; and
- `Pass_pass`: In this data set, the train will pass through both the origin and destination stations without calling at either of them.

(10) `congestion`: This attribute is given a double precision value, which represents the congestion rate of the train. Congestion rate is the number of passengers actually on board a train divided by the rated passenger capacity of the train.

(11) `station_object_number`: This attribute is given an integer value, which represents the number of station object associated with this `nextsta` data set. This part of RTSS is not actively maintained or supported.

(12) `nextsta_switch_point`: This attribute is given a double precision value, which represents the position in [km] of a place where `nextsta` data set is switched to the next one. When a train uses this `nextsta` data set and reaches

this position, it will switch to the next data set. This attribute is used when it is necessary to change from one `feedline` object under which the train run to another during the inter-station run.

Note that, when this attribute is set in one `<nextsta>` element, the `<nextsta>` element that follows it must have the same data for all child elements and all attributes except attributes `feeder_name` and `nextsta_switch_point`.

(13) `soc_feedforward_data`: This attribute is given a string value, which represents the name of the SOC curve data associated with this train when it runs using this `nextsta` data set. The named SOC curve data must be defined in the accompanying file; otherwise it will cause an error.

4.6 S File

(To be completed.)

4.7 P File

(To be completed.)

4.8 F File

(To be completed.)

5 Invoking RTSS

To invoke RTSS using the data files you have created, you must start the shell and run the following command:

```
rtss control_file_name.xml
```

This will cause RTSS to read the control file first, interpret it and find the appropriate files for reading.

If the binary executable file does not reside on one of the directories listed in the "Path" environment variable, you can invoke RTSS by typing in the full path to the binary executable file. (Note that the Windows installer will not set the environment variable – you must do it by hand if you want to.)

The control file name can be either a full path to it or the path relative to the current "working directory".

Index

A

C

D

E

F

G

L

N

P

R

S

T