VOCALOID™

Job Plug-in API Reference Manual

Article ID: VJP-1.0.0.3 / API Version: 3.0.1.0

Revision History

Article ID	API Version	Changes
VJP-1.0.0.0	3.0.0.1	First edition created
VJP-1.0.0.1	3.0.0.1	TIPS Add - How to use file input and output library function - How to run a command or external program
VJP-1.0.0.2	3.0.1.0	Added update function and acquisition of phonetic symbols protect note.
VJP-1.0.0.3	3.0.1.0	Execution environment parameter table to be passed to the main function of the Job plug-in script (the second parameter), to add a field that indicates the version of the Job plug-in API of current. API Add - I get a pre-major - I to get the audio device name Add the Job plug-in API list TIPS Add - How to determine the available Job plug-in API

1 Introduction

1.1 Job plug-ins and

Job plugin provides a mechanism capable of the script written in (Ver. 5.1.4) programming language Lua, to expand freely according to the needs of the user from the outside, a Musical part editing VOCALOID3 Editor thing. And this mechanism is provided by the Job plug-in API.
Job plug-in API, which consists of providing Musical part information update function and access to the Musical Part information from Lua script, the set of functions. Incidentally, to be referred to as Job the plug-in script, and the script after Lua, written as Job plug in this document.

1.2 Version

Version of the Job plug-in API that is described in this document is 3.0.1.0.

1.3 What someone can and can not do in the Job plug-in API

1.3.1 What you can do with Job plug-in API

You can by using the Job plug-in API.

- Retrieve and update of basic properties of notes Musical Part
- Retrieve and update of facial control parameters of notes Musical Part
- Retrieve and update vibrato type of notes Musical Part
- Retrieve and update vibrato lengths notes Musical Part
- Retrieve and update the control parameters of the Musical Part
- Retrieve and update the properties of Musical Part
- Get of the tempo of the master track
- Get of the time signature of the master track
- Get the properties of the sequence
- Get properties of and WAV part

1.3.2 It can not be in the Job plug-in API

Currently, we do not disclose the Job plug-in API to do the following.

- Retrieve and update vibrato control parameters of notes Musical Part
- Updating the tempo of the master track
- Updating the time signature of the master track
- Update the properties of a sequence
- Update the properties of a part WAV
- Retrieve and update the information other than those listed above

2 Structure of Job plug-in scripts

2.1 Syntax

Job plug-in script, conforms to the Syntax of language Lua. Also, I can access the language features of all the Lua language provides.
Remarks about the Syntax of Lua language will be beyond the scope of this document. Please refer to the manual of the commercial and official WEB page of the Lua language.

2.2 Entry point function

Job plug-in script must have a function named "main" always. This function will be used as the entry point of the Job plug-in script. In addition, the end of this function is the end of the entire Job plug-in.

2.2.1 Parameter list

"main" function has two parameters.

Is a table type variable that contains the time of the current song position of the target Musical part, and time of the start and end points selected by the user in the editor, the first parameter is given from VOCALOID3 editor. The unit is Tick which is 0 for the first part of Musical both. Your name and if you run the Job plug-in user can not select anything, the beginning of the part (0), the starting point will be at the end of the end time events that Part will end.

The Job plug-in, you will be able to perform the process of updating the data to take into account the range of time and end this starting point. In addition, the Get range of the data is the Musical part whole, because it does not limit the VOCALOID3 editor side also updatable range, it is also possible to perform the update is more than the above range. However, updating of data significantly above this range are possible results when different from the intention of the user who performed this. Therefore, the update of in excess of this range, you must make the implementation of consideration to fit within reasonable limits, for example, is acceptable as a margin of fade-in / fade-out, etc…

On the other hand, if you want to use like any this parameter, certain degree of freedom is allowed and depending on the processing of the Job each plugin. For example, I will consider using the following method.

○ For a plug-in to delete the specified section (Example: DeleteSelection.lua)

The time deleting section all events to define a deletion interval interval time of a start point and an end point selected by the user VOCALOID3 editor, delete all events that are present in the section, is present at the time of the section after I shifted forward only.

The Job plugin to the specifications of the above, data is updated beyond the period of a start point and an end point selected by the user, but if indicated to the user that specification, it is possible to create a Job such plugins I can be thought of as no problem.

○ If the Job plug-in to change the lyrics of notes specified time after

In this case, it could also be implemented as a Job plug-in specification is not used and time of start and end points selected by the user in the VOCALOID3 editor, is to change the lyrics note from the time of the song position. Also in this case, it is likely that if you specify to the user specification, and there is no problem that you want to create a Job such plug-ins.

The second parameter is a table type variable that contains information about the operating environment of the Job plug-in, the following information is included.

○ Path to the directory where the source code for the Job plug-in is invoked is disposed

The full path to the Job plug-in script files that are marked with delimiter "\" at the end. In addition, this path is the current directory of Job plug-in runtime.

○ Source file name Job plug was started

It is only Job plug-in script file name with the extension, it is not a full path。

○ The path of the temporary directory Job plug-in is available

It is the full path marked with delimiter "\" at the end. Job plug-ins, you can create a temporary directory and temporary files freely under this directory.

○ Version of current Job plug-in API

Is represented sets of four integer separated by ".", the version of current Job plug-in API. It should be noted that the Get of the above parameters other than the Job plug-in run-time, I will do through the Job plug-in API.

2.2.2 Return code

"main" function must return a return code of exactly one.

Return code is shown below.

○ It returns 0 if you want to apply to the Musical part updates from the script to terminate normally.

○ Returns a non-zero value if you want to cancel the update from the script or error.

There is no agreement about the value of the return code in the case of non-zero.

You can decide freely to the Job each plug-in. However, it is not possible to control the behavior of the editor

VOCALOID3 side by the value.

2.2.3 Example Code

I shows a sample entry point function of the simplest Job plug-in script below, do nothing.
(Example: DoNothing.lua)

```
--
-- Job plug-in script that does nothing.
--
-- Entry point function of the Job plug-in script.

function main(processParam,  envParam)
-- Get of the parameters passed to the runtime.
beginPosTick      = processParam.beginPosTick      -- Start time of the selection.
endPosTick        = processParam.endPosTick        -- End time of the selection.
songPosTick       = processParam.songPosTick       -- Current song position time.

-- I get the execution environment parameters passed at run time.
scriptDir         = envParam.scriptDir             -- Directory path where the script is located.
scriptName        = envParam.scriptName            -- File name of the script.
tempDir           = envParam.tempDir               -- Temporary directory path Job plug-in is available.
apiVersion        = envParam.apiVersion            -- Version of the Job plug-in API of current.

-- I do so now following the processing of the Job plug-in.
-- Successful completion.
return 0

end
```

2.3 Job plug-in manifest function

2.3.1 The information provided

to VOCALOID3 editor plug-in from the Job script side, or a thing what the Job plug-in, "manifest" function is a function to provide the information producer, version, etc.. Same function, "main" Job plug-in script, it must be equipped with all means this function.
This value function should return is as follows. Both can not be omitted.

- The name of the plug-in Job
- Comment describing the Job or plug-ins do anything
- Maker's name
- ID that uniquely identifies the Job plug-in (GUID)
- version of the Job plug-in (Version of the assembled four it has been joined with a dot)
- Version of the Job plug-in API to use ("3.0.1.0" at the moment)

In addition, ID that uniquely identifies the Job plug-in the above (GUID), please be generated using the guidgen.exe etc. that comes with the Microsoft Windows SDK. In this case, I want to registry format the format of the GUID to be generated.

2.3.2 Example Code

Below, I shows a sample plug-in manifest function of the Job plug-in script.

```
--
-- Plug-in manifest function.
--
function manifest()
myManifest = {
        name            = " God Torture plug-in",       -- The name of the Job plug-in
        comment= " And God Torture",                    -- Comment describing the Job plug-ins or do anything
        author          = "Plug-in craftsman ",         -- Maker's name
        pluginID        = "{AE31E35D-2FD4-4608-8D67-C2B45353192A}",     -- Job plug-in ID
        pluginVersion   = "1.0.0.1",                    -- Version of the Job plug-in
        apiVersion      = "3.0.0.1"                     -- Version of the Job plug-in API to be used
}
return  myManifest
end
```

3 Job plug-in API

3.1 Terms

3.1.1 Naming conventions

Job plug-in script, you can access the language features of all the Lua language provides. Therefore, in order to be seen clearly and avoid collisions with these names and is a Job Plugin API, and then apply the following naming convention is the Job plugin API.

○ Prefix of the API name

The Job plug-in API name, and give it the prefix "VS".

○ Prefix of Lua table name

Also table name of Lua table, and give it the prefix "VS".

○ Variable name

Variable name of the field name of the Lua table, begin with a letter in lower case.

3.1.2 Data type

There is no data type is basically the Lua language.

However, in order to maintain the integrity of the data VOCALOID3 editor handles, define the following data types for the sake of convenience here, you can choose to the following description using these data types in this document.

○ VSInt32

Is a 32-bit signed integer.

○ VSFloat

Single-precision floating-point number.

○ VSBool

Is a boolean value. When VS_TRUE (1) true, I take VS_FALSE (0) when false.

○ VSCString

Is a constant string. Available character code is only UTF-8.

3.1.3 How to define a table type

There is a data structure called a table type in Lua language.

On the Job plug-in API, I used to deal with structure of Musical part data (note information, etc.) table type.

This document describes using a pseudo-grammar structure similar definition of the C + + language as how to define a table type for the sake of convenience.

For example, the definition of the note event type is described as follows.

```
// Note event type.
struct VSLuaNote {
    VSInt32         posTick;        -- Notes ON time
    VSInt32         durTick;        -- Notes Duration
    VSInt32         noteNum;        -- Tone
    VSInt32         velocity;       -- Velocity
    VSCString       lyric;          -- Lyrics
    VSCString       phonemes;       -- phonetic symbols (Separated by spaces)
};
```

And, I will treat the following as a Lua table in the Job plug-in script this.

```
-- Set of note events.
note = {}                                   -- Generation of note table data.
note.posTick        = 1920                  -- Set of note ON time.
note. durTick       = 480                   -- Set of note Duration.
note.noteNum        = 69                    -- Setting the Tone.
note.velocity       = 64                    -- Set of velocity.
note.lyric          = “spring”              -- Set of lyrics.
note.phonemes       = "s p r I N"           -- Set of phonetic symbols.
```

3.1.4 How to define prototype of the Job plug-in API

Prototype definition of the Job plug-in API also uses a pseudo-grammar of C + + language similarity.
For example, in the Lua language, you can function return multiple values. Purototai of such API
I proceed as follows: In this document, the definition of a flop.

```
// I get an integer value that is input from the field of parameter input dialog.
// Parameters:
// fieldName: Field name in the dialog from which to retrieve.
//  Returns:
//  result: VS_TRUE when you have successfully retrieved, When VS_FALSE of error.
//  value:  Integer value that is input from the field of parameter input dialog.

VSBool result, VSInt32 value  =  VSDlgGetIntValue( VSCString  fieldname );
```

The above API, which means that it is an API that takes a string argument of one, returns the return code of two 32-bit signed integer and Boolean type.
Use a real example of the Job plug-in script of this API is as follows.

```
 -- I want to get the input value of the pitch of the note from the parameter input dialog.
 result, noteNum  =  VSDlgGetIntValue("noteNum")
```

Get the input value of 32-bit signed integer type from a field named "noteNum" of parameter input dialog, this example is stored in the variable named noteNum its value. To a variable of Boolean type called result, this API has contains the results of whether or not successful at the same time.

3.1.5 Character encoding

Character encoding that can be used in the Job plug-in script is only UTF-8 .

3.2 Passing API parameters and VOCALOID3 editor

3.2.1 Dynamic generation of parameter input dialog

According to parameter passing API call from the Job plug-in script, VOCALOID3 editor, dynamically generates a parameter input dialog to the Job plug-in script.

3.2.2 And Constant Definitions table type to use

The following are the constant definitions and table type definition to be used in the parameter passing API with VOCALOID3 editor.

```
// Field type of the parameter input dialog.
enum VSFlexDlgFieldType {
    FT_INTEGER = 0,                 // Integer type.
    FT_BOOL,                        // Boolean type.
    FT_FLOAT,                       // Real type.
    FT_STRING,                      // String type.
    FT_STRING_LIST                  // String type list (combo box).
};

// Field definition structure of the parameter input dialog.
struct VSFlexDlgField {
    VSCString       name;           // Field name.
    VSCString       caption;        // Caption of the field.
    VSCString       initialVal;     // The initial value of the field.
    VSInt32         type;           // Field Type  (I take the value of FlexDlgFieldType)
};
```

3.2.3 Adding API field definitions

○ VSBool VSDlgAddField(VSFlexDlgField field)

Add the input field information to be displayed to the parameter input dialog.

In addition, if you specify "FT_STRING_LIST" field type, field initialVal

The call by setting the string that are separated by commas (",") to. This combo box field is added to the parameter input dialog, it is inserted in units of strings separated by commas to each row in the combo box.

```
// I want to add a field to the parameter input dialog.
// Parameters:  Field definition of the parameter input dialog
// Returns: When VS_TRUE of success, for VS_FALSE of error.

VSBool  VSDlgAddField ( VSFlexDlgField  field );
```

○ void VSDlgSetDialogTitle (VSCString dlgTitle)

Returns: When VS_TRUE of success, for VS_FALSE of error.

```
// I set the window title of the parameter input dialog.
// Parameters: Window title string for the parameter input dialog.
// Returns: None.

void VSDlgSetDialogTitle ( VSCString  dlgTitle);
```

3.2.4 Instruction to display of the dialog API

○ VSInt32 VSDlgDoModal()

I will display the parameter input dialog.

In order to display as a modal dialog, control does not return OK or Cancel button until you press.

```
// To perform the input of parameters to display a modal dialog.
// Parameters: None
//Returns: When button OK pressed IDOK (1), when Cancel button pressed IDCANCEL (2).

VSInt32 VSDlgDoModal();
```

3.2.5 API get of the input parameters

○ VSBool result, VSInt32 value VSDlgGetIntValue(VSCString fieldName)

I get an integer value that is input from the field of parameter input dialog.

```
// I get an integer value that is input from the field of parameter input dialog.
// Parameters:
//          fieldName: Field name in the dialog from which to retrieve.
// Returns:
//          result: When acquisition is successful VS_TRUE, when the error VS_FALSE.
//          value:  Integer value that is input from the field of parameter input dialog.

VSBool result, VSInt32 value  =  VSDlgGetIntValue ( VSCString  fieldName );
```

○ VSBool result, VSBool value VSDlgGetBoolValue(VSCString fieldName)

I get a boolean value which is input from the field of parameter input dialog.

```
// I get an boolean  value that is input from the field of parameter input dialog.
// Parameters:
// fieldName: Field name in the dialog from which to retrieve.
// Returns:
// result: When acquisition is successful VS_TRUE,  when the error VS_FALSE.
// value:  Boolean input from the field of parameter input dialog.

VSBool result, VSBool value  =  VSDlgGetBoolValue ( VSCString  fieldName );
```

○ VSBool result, VSFloat value VSDlgGetFloatValue(VSCString fieldName)

I get a floating-point value that is input from the field of parameter input dialog.

```
// I get a floating-point value that is input from the field of parameter input dialog.
// Parameters:
// fieldName: Field name in the dialog from which to retrieve.
// Returns:
// result: When acquisition is successful VS_TRUE,  when the error VS_FALSE.
// value: Floating-point value that is input from the field of parameter input dialog.

VSBool result, VSFloat value VSDlgGetFloatValue ( VSCString  fieldName );
```

○ VSBool result, VSCString value VSDlgGetStringValue (VSCString fieldName)

I get a string value that is input from the field of parameter input dialog.

For fields that are specified "FT_STRING_LIST" field type, string stored in the selected row in the combo box is retrieved.

```
// I get a string value that is input from the field of parameter input dialog.
// Parameters:
// fieldName: Field name in the dialog from which to retrieve.
// Returns:
// result: When acquisition is successful VS_TRUE,  when the error VS_FALSE.
//  value:  String value input from the field of parameter input dialog.

VSBool result, VSCString value VSDlgGetStringValue ( VSCString fieldName );
```

3.2.6 Example program

○ DialogSample.lua

3.3 Get and edit of note events

3.3.1 Definition of table type to use

The following is a table type definition to be used in the get and editing of note events

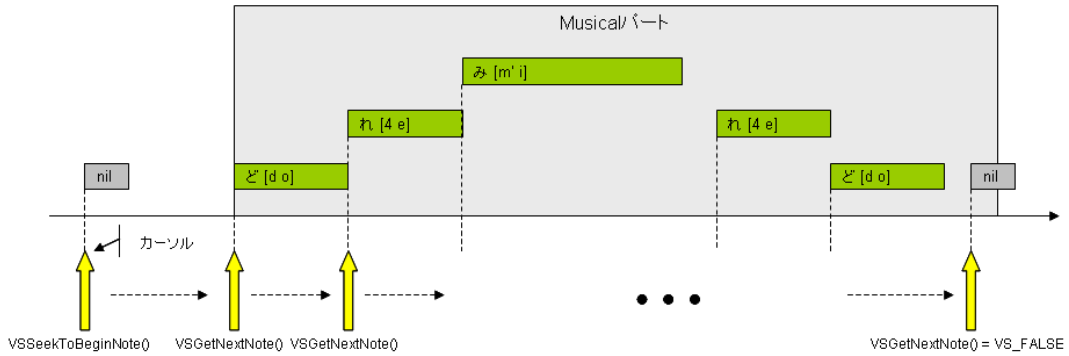
```
// Note event type.
struct VSLuaNote {
    VSInt32         posTick;        // Note ON time
    VSInt32         durTick;        // Note Duration
    VSInt32         noteNum;        // Tone
    VSInt32         velocity;       // Velocity
    VSCString       lyric;          // Lyrics
    VSCString       phonemes;       // Phonetic symbols (It is separated by a space)
    VSBool          phLock;         // Phonetic protect flag (Optional default: VS_FALSE)
};
```

```
// Note event type, including the facial control parameters.
struct VSLuaNoteEx {
    VSInt32         posTick;        // Note ON time
    VSInt32         durTick;        // Note Duration
    VSInt32         noteNum;        // Tone
    VSInt32         velocity;       // Velocity
    VSCString       lyric;          // Lyrics
    VSCString       phonemes;       // Phonetic symbols (It is separated by a space)
    VSBool          phLock;         // Phonetic protect flag (Optional default: VS_FALSE)

    VSInt32         bendDepth;      // Bend Depth (0 - 100)
    VSInt32         bendLength;     // Bend Length (0 - 100)
    VSBool          risePort;       // Addition of portamento flag on line form
    VSBool          fallPort;       // Portamento additional flag in the descending form
    VSInt32         decay;          // Decay (0 - 100)
    VSInt32         accent;         // Accent (0 - 100)
    VSInt32         opening;        // Opening (0 - 127)
// Vibrato length (Percentages rounded to integer)
    VSInt32         vibratoLength;
// Vibrato type (Type of vibrato)
// 0:  Do not offend vibrato
// 1 - 4: Normal // 5 - 8: Extreme // 9 -12: Fast // 13 - 16: Slight
    VSInt32         vibratoType;
};
```

Here, phLock field was added in API version 3.0.1.0. This is a flag to protect the pronunciation symbol of the note. When you set the VS_TRUE, phonetic symbols of the note will be protected. When you set the VS_FALSE, protection of phonetic symbols of the note is released.
Protection of the phonetic symbols, are from automatic conversion feature of the pronunciation symbols to be executed when you edit the lyrics from VOCALOID3 editor, to protect the phonetic symbols of the note. It is not intended to protect against changes from the Job plug-in. Therefore, from the Job plug-in, you can change the phonetic symbols also to note that this protection is wound. Protected from over The decision to change to note that, you are left to the discretion of the Job plug-in depending on the use of the Job each plugin.

3.3.2 The sequential access by cursor

Access to the note events can be carried out by sequential access from the Musical part beginning with a cursor.
First, VSSeekToBeginNote() By, to position the cursor on the note virtual one before the first note of the Musical Part.
Then, you can VSGetNextNote by () API, and position the cursor to the top notes of Musical part, to get the value of the note. And later, by calling repeatedly VSGetNextNote the () API, you can get to the end notes Musical notes part of the following sequence.
If you call the VSGetNextNote () API to get after the end note of Musical part, this
Returns (0) VS_FALSE, API will only position under the notes virtual behind one of the end note the cursor.



The sequential access by cursor

○ void VSSeekToBeginNote()
I Position the notes a virtual one before the first notes of the Musical Part cursor.

```
// I positioned one before the first notes of the Musical Part cursor.
// Parameters: None.
// Returns: None.

void VSSeekToBeginNote() ;
```

○ VSBool result, VSLuaNote note VSGetNextNote()
To advance to the next note the cursor, you will get the value of the note.

```
// To advance to the next note the cursor, to get the value of the note.
// Parameters: None.
// Returns:
// result: When acquisition is successful VS_TRUE, When the error VS_FALSE.
// note: Value of the note.

VSBool result, VSLuaNote note  =  VSGetNextNote();
```

3.3.3 Update notes

○ VSBool VSUpdateNote(VSLuaNote note)

Update notes the value of the notes that was passed in the parameter.

notes that can be passed to the parameter, which must be a notes that you obtained in VSGetNextNote API in advance. In other words, get a note on VSGetNextNote API, set the value to be updated with respect to the field of the notes, the steps to update the value of the note by calling the API as a parameter of VSUpdateNote API it then.

```
// I will update the value of the notes.
// Parameters: notes that the updated values is set.
// result: When acquisition is successful VS_TRUE, When the error VS_FALSE.

VSBool VSUpdateNote(VSLuaNote note);
```

3.3.4 Add notes

○ VSBool VSInsertNote(VSLuaNote note)

You can add a notes with the value of the note you pass a parameter。

Each field of notes passed in the parameter must be set with the appropriate values. However, it can be omitted phLock field. If omitted, the default value

I will be set (not protect the phonetic symbols) VS_FALSE.

It is not possible to omit fields other than phLock.

```
// I add a note.
// Parameters: note to be added.
// Returned value: When acquisition is successful  VS_TRUE, then the error  VS_FALSE.

VSBool VSInsertNote( VSLuaNote note );
```

3.3.5 Delete the note

○ VSBool VSRemoveNote(VSLuaNote note)

You can remove the note that was passed in the parameter.

note that can be passed to the parameter, not becomes a note must be obtained in advance VSGetNextNote API. It is the procedure that is to say, to get the note Ki Delete all in VSGetNextNote API, to remove the note by calling the API as a parameter of VSRemoveNote API it.

```
// I delete the note.
// Parameters: deleting note.
// Returned value: When acquisition is successful  VS_TRUE, then the error  VS_FALSE.

VSBool VSRemoveNote(VSLuaNote note);
```

3.3.6 The API for facial control parameters of the note

○ VSBool result, VSLuaNoteEx noteEx VSGetNextNoteEx()

To advance to the next note the cursor, obtain the value of the note, including the facial control parameters of the note.

```
// To advance to the next note the cursor, to get the value of the note, including the facial control parameters.
// Parametrs: None
// Returned value:  When acquisition is successful  VS_TRUE, then the error  VS_FALSE.
// noteEx:  Value of the note, including the facial control parameters.

VSBool result, VSLuaNoteEx noteEx = VSGetNextNoteEx();
```

○ VSBool VSUpdateNoteEx(VSLuaNoteEx noteEx)
It updates the value of the note, including the facial control parameters you pass in the parameter. note that can be passed to the parameter, which must be a note containing the facial control parameters obtained in VSGetNextNoteEx () API in advance.

```
// I will update the value of the note, including the facial control parameters.
// Parameters: note, including the facial control parameters.
// Returned value:  When acquisition is successful  VS_TRUE, then the error  VS_FALSE.

VSBool  VSUpdateNoteEx( VSLuaNoteEx noteEx );
```

○ VSBool VSInsertNoteEx(VSLuaNoteEx noteEx)
Add a note containing the facial control parameters you pass in the parameter. Each field of notes passed in the parameter must be set with the appropriate values. However, it can be omitted phLock field. If omitted, it is set (not protect the phonetic symbols) VS_FALSE as the default value. It is not possible to omit fields other than phLock.

```
// I add a note containing the facial control parameters.
// Parameters: note, including the facial control parameters to be added.
// Returned value:  When acquisition is successful  VS_TRUE, then the error  VS_FALSE.

VSBool  VSInsertNoteEx( VSLuaNoteEx noteEx );
```

3.3.7 A sample program
○ NoteSample1.lua
○ NoteSample2.lua

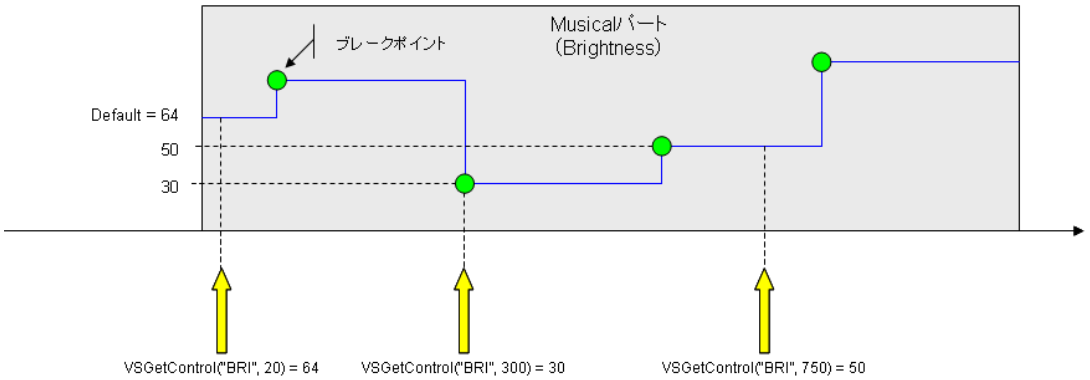
3.4 Obtaining and editing control parameters
3.4.1 Type and Constant Definitions of the table to use
Table type to be used in the Editing and API acquisition of control parameters and following shows the definition of the constant.

```
// Type of control parameter (string).
// "DYN": Dynamics
// "BRE": Bureshinesu
// "BRI": Brightness
// "CLE": Clearnes
// "GEN": Gender factor
// "PIT": Pitch
// "PBS": Pitch Bend Sensitivity
// "POR": Portamento timing

// Control parameters breakpoint type

struct VSLuaControl {
    VSInt32         posTick;    // Position of the control parameters breakpoint.
    VSInt32         value;      // Value of the control parameter breakpoint.
    VSCString       type;       // Type of control parameters.
};
```

3.4.2 The random access method specified by the time

Control parameter, has existed as a break point that is located discretely on the time axis is implementation, but I have a property to change continuously with respect to the time axis in concept. In this case, the access method of the data, it is likely than the access method, and easy to understand more of the random access method specified by the time more intuitive sequential like note.
Therefore, for the control parameters, and to be introduced by the random access method specified by time.
The figure below shows an example of using the method of this. The value of the control parameters, because between the break point and break point are interpolated by a straight line in the manner shown in the figure below, the value is always present at the time of any part in the Musical. For this reason, random access method specified by time is possible.

○ If you specify a time before the first break point
API returns the default value of the control parameter.
○ If you specify the time of the break point on
API returns the value of the break point.
○ If the break point-to-point and specify a time after the end breakpoint
API returns the value of the break point of the maximum time that existed before that time.



The random access method specified by the time

3.4.3 The random access method specified by the time

○ VSBool result, VSInt32 value VSGetControlAt(VSCString controlType, VSInt32 posTick)
I get the value of the control specified parameters of the specified time of the Musical Part.

```
// I get the control parameters.
// Parameters:
// controlType: Type of control parameters to be retrieved.
// posTick: Time of control the parameter to retrieve.
// Time in units of Tick in a 0 for the first part.
// Return value:
// result: When acquisition is successful VS_TRUE, When the error VS_FALSE.
// value:  Value of the control parameter.

VSBool result, VSInt32 value VSGetControlAt(VSCString controlType, VSInt32 posTick);
```

○ VSBool VSUpdateControlAt(VSCString controlType, VSInt32 posTick, VSInt32 value)
I will update the value of the control specified parameters of the specified time of the Musical Part.

```
// I will update the value of a control parameter.
//  Parameters:
// controlType: Type of control parameters to be retrieved.
// posTick: Time of control the parameter to retrieve.
// Time in units of Tick in a 0 for the first part.
// value: Value of the control parameter to be updated.
// Return value: When acquisition is successful VS_TRUE, When the error VS_FALSE.

VSBool VSUpdateControlAt( VSCString controlType, VSInt32 posTick, VSInt32 value );
```

3.4.4 The sequential access method API with Cursor

You may want also want to know for the control parameters, where to find the change point of the value or there. In such a case, better to while examining sequentially from the beginning break point of the control parameters is more efficient.

So, I will have a sequential access API with a cursor also to control parameters.

○ VSBool VSSeekToBeginControl(VSCString controlType)

I positioned before one of the first control parameter the specified breakpoints Musical part the cursor.

```
// The first control parameter the specified breakpoints Musical part the cursor
// I positioned one before.
// Parameters:
// controlType: Type of control parameters to be retrieved.
// Return value: When acquisition is successful VS_TRUE, When the error VS_FALSE.

VSBool VSSeekToBeginControl( VSCString controlType );
```

○ VSBool result, VSLuaControl control VSGetNextControl(VSCString controlType)

To advance to the control parameters the next breakpoint of the control parameters the specified breakpoints Musical part the cursor to retrieve its value.

```
// To advance to the control parameters breakpoint of the cursor to the next, to get its value.
// Parameters:
// controlType: Type of control parameters to be retrieved.
// Return value:
// result:  When acquisition is successful VS_TRUE, When the error VS_FALSE.
// control: Value of the control parameter breakpoint.

VSBool result, VSLuaControl control = VSGetNextControl(VSCString controlType);
```

○ VSBool VSUpdateControl(VSLuaControl control)

Update control events in the value of the control event you pass a parameter. Control events that can be passed to the parameter, which must be the ones that you obtained in VSGetNextControl API in advance. In other words, to get the control events in VSGetNextControl, set the value to be updated for the field of the control event, the steps to update the value of a control event by calling the API as a parameter of VSUpdateControl it then.

```
// I will update the value of a control event.
// Parameters: Control events that value to be updated has been set.
// Return value: When acquisition is successful  VS_TRUE, When the error  VS_FALSE.

VSBool VSUpdateControl( VSLuaControl control );
```

○ VSBool VSInsertControl(VSLuaControl control)
Add control events in the value of the control event you pass a parameter. Each field of control events that pass in the parameter must be set with the appropriate values. (You can not omit the field of all.)

```
// I want to add a control event.
//  Parameters: Control event to be added.
// Return value: When acquisition is successful VS_TRUE, When the error VS_FALSE.

VSBool VSInsertControl( VSLuaControl control );
```

○ VSBool VSRemoveControl(VSLuaControl control)
Remove the control event you pass a parameter. Control events that can be passed to the parameter, which must be the ones that you get in VSGetNextControl () API in advance. In other words, get the control events to be deleted VSGetNextControl in () API, it is the procedure to remove a control event by calling the API as a parameter VSRemoveControl of () API it.

```
// I want to remove a control event.
//  Parameters: Delete target control event.
// Return value: When acquisition is successful VS_TRUE, When the error VS_FALSE.

VSBool VSRemoveControl( VSLuaControl control );
```

3.4.5 Getting default value
○ VSBool result, VSInt32 value VSGetDefaultControlValue(VSCString controlType)
Gets the default value of the control parameter the specified type.

```
// Get the default value of the control parameter the specified type.
// Parameters:
// controlType: Type of control parameters to be retrieved.
// Return value:
// result: When acquisition is successful VS_TRUE, When the error VS_FALSE.
// value:  The default value of the control parameter.

VSBool result, VSInt32 value VSGetDefaultControlValue( VSCString controlType );
```

3.4.6 Sample program
○ ControlSample1.lua
○ ControlSample2.lua
○ ControlSample3.lua

3.5 Obtaining the Master track information

3.5.1 Definition of table type to use

The following is a table type definition to be used in the acquisition of API master track information. In addition, you need to be careful of the master track time information, because it is represented by a global Tick whose origin is the sequence head all. (Time covered by the Musical Part information is part Tick relative to the origin Musical head part.)

```
// Tempo event type.

 struct VSLuaTempo {
      VSInt32          posTick;          // Tempo Time (Global Tick).
      VSFloat          tempo;            // Tempo value (BPM).
};

// Time signature event type.

struct VSLuaTimeSig {
      VSInt32          posTick;          // Beat time (global Tick).
      VSInt32          numerator;        // Numerator of time signature.
      VSInt32          denominator;      // The denominator of the time signature.
};
```

3.5.2 The sequential access by cursor

Access to the tempo and time signature event event of the master track, can be done by sequential access from the sequence beginning with a cursor.

First, the VSSeekToBeginTempo API, I positioned the cursor to the tempo a virtual one before the beginning of the sequence tempo.

Then, you can by VSGetNextTempo API, and position the cursor to the top of the sequence tempo, to get the value of its tempo. And later, by the call to repeat the VSGetNextTempo () API, you can get to the end of the sequence tempo the tempo of the next sequence.

If you call the VSGetNextTempo () API to get after the end of the tempo of the sequence, returns VS_FALSE(0), this API is positioned in the tempo of virtual behind one of the end tempo cursor.

The above behavior is the same for the time signature event.

○ void VSSeekToBeginTempo()

I positioned before one of the first tempo of the sequence tempo cursor.

```
// I positioned before one of the first tempo of the sequence tempo cursor.
// Parameters: None.
// Returns: None.

void VSSeekToBeginTempo();
```

○ void VSSeekToBeginTimeSig()
I positioned before one of the first time signature of the time signature sequence cursor.

```
// I positioned before one of the first time signature of the time signature sequence cursor.
// Parameters: None.
// Returns: None.

void VSSeekToBeginTimeSig();
```

○ VSBool result, VSLuaTempo tempo VSGetNextTempo()
To advance to the store the next tempo cursor to retrieve the value of the tempo.

```
// To advance to the tempo of the cursor to the next, to get the value of the tempo.
// Parameters: None.
// Returns:
// result: When acquisition is successful VS_TRUE, When the error VS_FALSE.
// tempo:  Value of tempo.

VSBool result, VSLuaTempo tempo =  VSGetNextTempo();
```

○ VSBool result, VSLuaTimeSig timeSig VSGetNextTimeSig()
And proceeds to time signature the next time signature cursor to retrieve the value of that time signature.

```
// To advance to the time signature of the cursor to the next, to get the value of the time signature.
// Parameters: None.
// Returns:
// result: When acquisition is successful VS_TRUE, When the error VS_FALSE.
// timeSig: Value of time signature.

VSBool result, VSLuaTimeSig timeSig = VSGetNextTimeSig();
```

3.5.3 The random access time specified
Also access to the tempo and time signature event event of the master track, random access by the time specified will be available. How it works is similar to the random access time of the control to the specified parameters. However, the designation of the time of the master track, the only difference from that of the control parameter is that it is a global Tick whose origin is always a sequence head. Therefore, if you want to associate with the value of the control parameter in the Musical part in, you need to be careful conversion of the Musical part in local and global time since time is required.

○ VSBool result, VSFloat tempo VSGetTempoAt(VSInt32 posTick)
I get the value of the tempo of the specified time.

```
// I want to get the value of the tempo of the specified time.
// Parameters:
// posTick: Time of tempo to get.
// Time in units of Tick which is 0 to the beginning of the sequence.
// Returns:
// result: When acquisition is successful VS_TRUE, When the error VS_FALSE.
// tempo: Value of tempo (BPM).

VSBool result, VSFloat tempo = VSGetTempoAt( VSInt32 posTick );
```

○ VSBool result, VSInt32 numerator, VSInt32 denominator VSGetTimeSigAt(VSInt32 posTick)
I get the value of the time signature of the specified time.

```
// I want to get the value of the time signature of the specified time.
// Parameters:
// posTick: Time of time signature to get.
// Time in units of Tick which is 0 to the beginning of the sequence.
// Returns:
// result:  When acquisition is successful VS_TRUE, When the error VS_FALSE.
// numerator:    Numerator of time signature.
// denominator: The denominator of the time signature.

VSBool result, VSInt32 numerator, VSInt32 denominator  = VSGetTimeSigAt( VSInt32 posTick );
```

3.5.4 Information acquisition of the entire sequence

○ VSCString VSGetSequenceName()
I get the sequence file name that is currently open.

```
// I get the sequence file name.
// Parameters: None.
// return value: file name of the currently open Sequence.

VSCString VSGetSequenceName();
```

○ VSCString VSGetSequencePath()
I get the sequence file path that is currently open.

```
// I get a sequence file path.
// Parameters: None.
// Returns:  Sequence file path that is currently open.

VSCString VSGetSequencePath();
```

○ VSInt32 VSGetResolution()
You will get a time resolution of the sequence that is currently open.

```
// I get the resolution time.
// Parameters: None.
// Returns:  Time resolution of the sequence that is currently open.

VSInt32 VSGetResolution();
```

○ VSInt32 VSGetPreMeasure()
I get a pre-measure of sequence that is currently open. The unit is bar.

```
// I get a pre-measure.
// Parameters: None.
// Returns:   Pre-measure. Units of the sequence currently open bar.

VSInt32 VSGetPreMeasure();
```

○ VSInt32 VSGetPreMeasureInTick()
Pre-measure of sequence that is currently open, I get as a unit the Tick. This means the start time of first bar.

```
// I get a pre-measure.
// Parameters: None.
// Returns:   Pre-measure. Units of the sequence currently open Tick.

VSInt32 VSGetPreMeasureInTick();
```

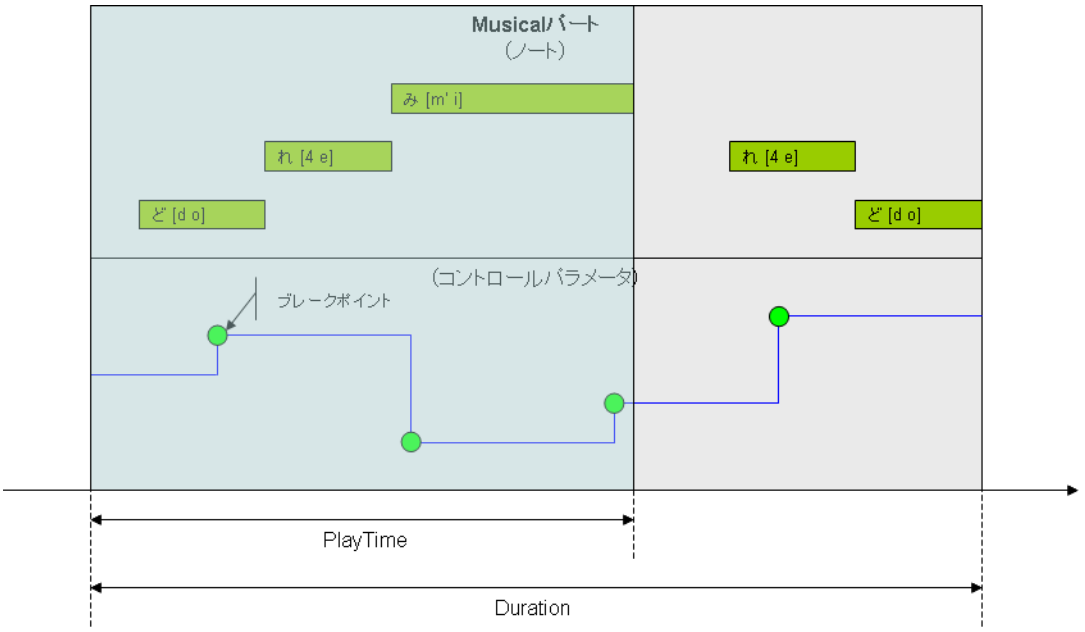
3.5.5 A sample program
○ MasterTrackSample.lua

3.6 Obtain and editing of Musical part information

3.6.1 Definition of table type to use
The following is a table type definition to be used in the editing and API acquisition of Musical part information.

```
// Part type.
struct VSLuaMusicalPart {
        VSInt32             posTick;            // Position of the part (Global Tick).
        VSInt32             playTime;           // Maximum duration of the part.
        VSInt32             durTick;            // Duration of the part.
        VSCString           name;               // Part name.
        VSCString           comment;            // Comment.
};

// Virtual Singer type
struct VSLuaMusicalSinger {
        VSInt32             vBS;                // Virtual Bank Select
        VSInt32             vPC;                // Virtual program change.
        VSCString           compID;             // Component ID.
// Sound parameters.
        VSInt32             breathiness;        // Bureshinesu
        VSInt32             brightness;         // Brightness
        VSInt32             clearness;          // Clearness
        VSInt32             genderFactor;       // Gender factor
        VSInt32             opening;            // Opening
};
```

3.6.2 About PlayTime And Duration
Including the Musical part, data, which means the length of two PlayTime and Duration will exist in the part.
PlayTime Means (Tick) time that is actually played.
Duration means the distance between the end time of the event and the end part time head of the part. Units, Tick the Musical part, For WAV part is Sec. In the figure below, I show the relationship PlayTime and Duration.



PlayTime and Duration Relationship

3.6.3 Information acquisition part of the Musical

○ VSBool result, VSLuaMusicalPart part VSGetMusicalPart()

I get the Musical part of the information currently processed.

```
// I get the Musical part of the information currently processed.
// Parameters: None.
// Returns:
// result: When acquisition is successful VS_TRUE, When the error VS_FALSE.
// part: Musical part of the information currently processed.

VSBool result, VSLuaMusicalPart part = VSGetMusicalPart();
```

3.6.4 Update information of Musical Part

○ VSBool VSUpdateMusicalPart(VSLuaMusicalPart part)

Update the Musical part by the value of the Musical part you pass a parameter.

Musical part that can be passed to the parameter, which must be a Musical part that you obtained in VSGetMusicalPart in advance. In other words, set the value to get the Musical part in VSGetMusicalPart, Product will all be updated with respect to that field, the value of the Musical part by calling the API as a parameter of VSUpdateMusicalPart the Re after its its it is the procedure to update.

```
// I will update the Musical part of information currently processed.
// Parameters: Musical Part information to be updated.
// result: When acquisition is successful VS_TRUE, When the error VS_FALSE.

VSBool VSUpdateMusicalPart( VSLuaMusicalPart part );
```

3.6.5 Virtual Singer information acquisition of the Musical part

○ VSBool result, VSLuaMusicalSinger singer VSGetMusicalPartSinger()

I get the information of virtual Singer Musical part of the current process target. Virtual Singer of Musical part is present only always one part Musical.

```
// I get the information of virtual Singer Musical part of the current process target.
// Parameters: None.
// Returns:
// result: When acquisition is successful VS_TRUE, When the error VS_FALSE.
// singer: Virtual Singer Musical part of the information to be processed currently.

VSBool result, VSLuaMusicalSinger singer = VSGetMusicalPartSinger();
```

3.6.6 A sample program
MusicalPartSample.lua

3.7 Get information of WAV part

3.7.1 Definition of table type to use
The following is a table type definition to be used in the acquisition of the WAV part information.

```
// WAV part type.
struct VSLuaWAVPart {
    VSInt32         posTick;        // Position of the part (Global Tick)
    VSInt32         playTime;       // Maximum duration of the part
    VSInt32         sampleRate;     // Sampling frequency
    VSInt32         sampleReso;     // Bit depth
    VSInt32         channels;       // Number of Channels
    VSCString       name;           // Part name.
    VSCString       comment;        // Comment
    VSCString       filePath;       // Absolute path of the WAV file
};
```

3.7.2 WAV part information obtaining
○ VSBool result, VSLuaWAVPart wavPart VSGetStereoWAVPart()
I get a WAV part information of a stereo track.
Because you do not have more than one maximum WAV part of a stereo track, you will get a WAV part of the only one in this API. (If the WAV part is present in the stereo track)

```
// I get a WAV part information of a stereo track.
// Parameters: None.
// Returns:
// result:  When acquisition is successful VS_TRUE, when the errorVS_FALSE.
// wavPart: WAV part information of a stereo track.

VSBool result, VSLuaWAVPart wavPart = VSGetStereoWAVPart();
```

○ void VSSeekToBeginMonoWAVPart()
I positioned before one of the top part WAV, WAV the cursor part of mono tracks。

```
// I positioned before one of the first part the WAV WAV cursor part of mono tracks.
// Parameters: None.
// Returns: None.

void VSSeekToBeginMonoWAVPart();
```

○ VSBool result, VSLuaWAVPart wavPart VSGetNextMonoWAVPart()
To advance to the next part of WAV to WAV cursor part of mono track, you will get the WAV part information.

```
// To advance to the next WAV part cursor of mono tracks, to get the WAV part information.
// Parameters: None.
// Returns:
// result:  When acquisition is successful VS_TRUE, when the errorVS_FALSE.
// wavPart: WAV part information of mono tracks.

VSBool result, VSLuaWAVPart wavPart = VSGetNextMonoWAVPart();
```

3.7.3 A sample program
○ WAVPartSample.lua

3.8 Status display in progress

○ VSInt32 VSMessageBox(VSCString message, VSUint32 type)
Show me to the message in a message box.

```
// I want to display a message box to messages.
// Parameters:
// message: String to be displayed.
// type: Style of the message box.
// MB_OK         = 0
// MB_OKCANCEL = 1
// MB_ABORTRETRYIGNORE = 2
// MB_YESNOCANCEL = 3
// MB_YESNO = 4
// MB_RETRYCANCEL  = 5
// Return value: Result code from the message box.
// IDOK = 1
// IDCANCEL = 2
// IDABORT  = 3
// IDRETRY  = 4
// IDIGNORE = 5
// IDYES = 6
// IDNO = 7

VSInt32 VSMessageBox(VSCString message, VSUint32 type);
```

3.9 Get of audio device information

○ VSCString VSGetAudioDeviceName()
Get the device name of the audio device you are currently using.

```
// I get the device name of the audio device currently in use.
// Parameters: None
// Returns: Device name of the audio device currently in use

VSCString VSGetAudioDeviceName();
```

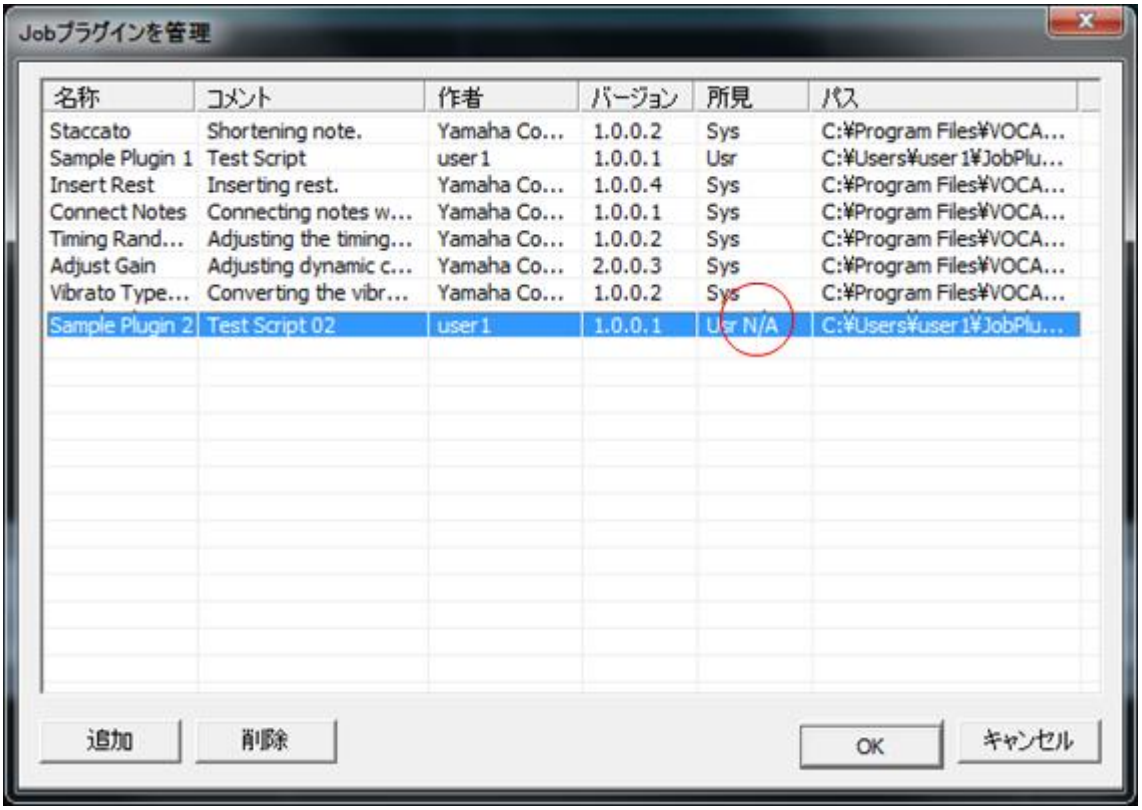
4 Registration the Job plug-in and Modification of the Job plug-in scripts

4.1 Registration of Job plug-in

"Job" of VOCALOID3 editor - By selecting> the menu, "Managing the Job Plug-in", dialog to manage the Job script plug-in opens. The dialog, the list of Job plug-in scripts that are currently registered will be displayed. From here, I can delete the registered Job plug-in script and add new Job plug-in script. You can press on the dialog "Add" button, you can add by file selection dialog appears, select the Job plug-in script file that you want to add here.

Here, I'll explain below an overview like any additional processing of the Job plug-in script, if it is done in VOCALOID3 within the editor.

First, load the Job plug-in a specified script file, you will get a plug-in manifest Job of the Job plug-in script. Then, only if it verifies the Job plug manifest acquired, it is determined to be a legitimate Job plug, register for Job plug library VOCALOID3 editor is managing. At this time, we calculate the SHA-1 hash code of the Job plug-in script file to register to Job plug-in library Te Align well.



Job plug-in script management dialog

4.2 Modification of the Job plug-in scripts

VOCALOID3 editor, has been managed by the internal database will keep in Job plug-in library, information Job plug-in registered.

Then, if the SHA-1 hash code of the Job plug-in script file, which is calculated at the time of registration of the Job plug-in if you have changed by subsequent editing, security measures on, VOCALOID3 editor is performed as if "falsification of malicious" across the board it is based on the state that it can not.

In this case, marked as "N / A" will be put in the Job plug-in script management dialog box in the "findings" field, you can see that it is ready to the Job plug-in can not be executed by edited by this.

So, you need a procedure if you have modified the Job plug-in script after registering the Job plug-in, remove it from the Job plug-in script management dialog first, and to register again.

It should be noted that the deletion of the Job plug-in is done by a user to choose from a list of Job plug-in purposes from the Job script plug-in management dialog and press the "Delete" button.

5 Job plug-in API list

API Name	Support API version
VSDlgSetDialogTitle	3.0.0.1 ～
VSDlgAddField	3.0.0.1 ～
VSDlgGetIntValue	3.0.0.1 ～
VSDlgGetBoolValue	3.0.0.1 ～
VSDlgGetFloatValue	3.0.0.1 ～
VSDlgGetStringValue	3.0.0.1 ～
VSDlgDoModal	3.0.0.1 ～
VSMessageBox	3.0.0.1 ～
VSGetSequenceName	3.0.0.1 ～
VSGetSequencePath	3.0.0.1 ～
VSGetResolution	3.0.0.1 ～
VSGetPreMeasure	3.0.1.0 ～
VSGetPreMeasureInTick	3.0.1.0 ～
VSSeekToBeginTempo	3.0.0.1 ～
VSGetNextTempo	3.0.0.1 ～
VSSeekToBeginTimeSig	3.0.0.1 ～
VSGetNextTimeSig	3.0.0.1 ～
VSGetTempoAt	3.0.0.1 ～
VSGetTimeSigAt	3.0.0.1 ～
VSGetMusicalPart	3.0.0.1 ～
VSUpdateMusicalPart	3.0.0.1 ～
VSGetMusicalPartSinger	3.0.0.1 ～
VSSeekToBeginNote	3.0.0.1 ～
VSGetNextNote	3.0.0.1 ～
VSGetNextNoteEx	3.0.0.1 ～
VSUpdateNote	3.0.0.1 ～
VSUpdateNoteEx	3.0.0.1 ～
VSInsertNote	3.0.0.1 ～
VSInsertNoteEx	3.0.0.1 ～
VSRemoveNote	3.0.0.1 ～
VSGetControlAt	3.0.0.1 ～
VSUpdateControlAt	3.0.0.1 ～
VSSeekToBeginControl	3.0.0.1 ～
VSGetNextControl	3.0.0.1 ～
VSUpdateControl	3.0.0.1 ～
VSInsertControl	3.0.0.1 ～
VSRemoveControl	3.0.0.1 ～
VSGetDefaultControlValue	3.0.0.1 ～
VSSeekToBeginMonoWAVPart	3.0.0.1 ～
VSGetNextMonoWAVPart	3.0.0.1 ～
VSGetStereoWAVPart	3.0.0.1 ～
VSGetAudioDeviceName	3.0.1.0 ～

6 TIPS

6.1 How to use file input and output library function

As mentioned in section 3.1.5, character code that can be used in the plug-in script Job is only UTF-8. For this reason, you need to be careful when passing to file input and output library function of Lua, the file paths that contain non-ASCII characters, such as Japanese.
More specifically, you need to be careful if you are using a file input and output library functions next.

- io.open()
- io.input()
- io.output()

○ If you want to pass the file path obtained from the parameter input dialog
String obtained by VSDlgGetStringValue () from the parameter input dialog, are encoded in UTF-8 in advance. Therefore, this string can be passed to the file input and output library functions as a file path as it is.
I show how you can use the following example.

```
-- I want to add a field to the parameter input dialog.
field = {}
field.name = "outFilePath"
field.caption  = " Output file path"
field.initialVal = ""
field.type = 3

dlgStatus = VSDlgAddField(field)

-- I get the output destination file path from the parameter input dialog.
dlgStatus = VSDlgDoModal()
dlgStatus, outFilePath = VSDlgGetStringValue("outFilePath")

-- File output.

outFile, errMsg = io.open(outFilePath, "w")

if (outFile) then outFile:write("Hello, World!!\n") end
```

○ If you want to pass the file path directly
As mentioned in section 3.1.5, Job plug-in script must be saved as a file encoded in UTF-8. And, as long as it is encoded in UTF-8, you can be passed to the file input and output library function as a literal directly, the file path-ASCII characters or other non-Japanese.
I show how you can use the following example.

```
function main(processParam, envParam)
-- Is assigned to the output file path that you specify.
io.output(envParam.tempDir .. "\\Hello.txt")

-- I output to a file.
io.write("Hello,World!!\n")
return 0
end
```

6.2 How to run a command or external program

You can plug-in from the Job script to run DOS commands and external programs. Here, I will explain the treatment method.
Description of this section, applies to Lua library functions below.

- os.execute()

○ Handling of the path of an external program
Available character code UTF-8 is the only path as an external program. In addition, the handling is the same as Section 5.1 above.
I show how you can use the following example.

```
-- I start Notepad.
os.execute("notepad.exe")
```

※ os.execute () function does not return control external program you start to finish.

○ How to run DOS command
If you want to run a DOS command, passing in the command string: as an argument os.execute () function.
CMD /C "DOS Command"

※ I must be enclosed in double quotes DOS command。
I show how you can use the following example.

```
-- The path of the directory to be created.
newDir = "C:\\Temp\\myDir"

-- I want to create a directory.
-- CMD /C "MKDIR "C:\Temp\myDir""

os.execute("CMD /C \"MKDIR \"" .. newDir .. "\"\"")
```

6.3 How to determine the available Job plug-in API

Job plug-in API is stored in a table that contains a list of global variable called "_G" of Lua. Thus, by defining the Job plug-in script functions such as the following can be used to determine the available Job plug-in API in the environment that is currently running.

```
-- I examine Job plug-in API is whether available.
-- Argument  apiName: Job plug-in API name to find out
-- Return value available: true  Not available: false
function isAvailableAPI(apiName)
  local isAvailable = false
  local key, val
  for key, val in pairs(_G) do
    if ((key == apiName) and (type(val) == "function")) then
       -- apiName Functions that match in there (available).
      isAvailable = true
      break
    end
end

return isAvailable
end
```

I use, for example: this function.

```
function main(processParam, envParam)
-- I get the name of a sequence.
sequenceName = ""
if (isAvailableAPI("VSGetSequenceName")) then
  sequenceName = VSGetSequenceName()
else
  -- And exit because you can not use the name of a sequence to get API.
  VSMessageBox("VSGetSequenceName is NOT available.", 0)
  return 1
end
-- Main routine.
status = doMain()
return status
end
```