

Json Library Reference

© 2014 Sony Computer Entertainment Inc.
All Rights Reserved.
SCE Confidential

Table of Contents

Introduction.....	7
Library Summary	8
sce::Json Namespace.....	9
Summary	10
sce::Json	10
Constants	11
Return Codes	11
Enumeration Types	12
ValueType	12
sce::Json::String Class	13
Summary	14
sce::Json::String.....	14
Constructors and Destructors	16
String	16
String(String)	17
String(char).....	18
~String	19
Public Class Methods.....	20
append(String).....	20
append(char).....	21
append(char,length)	22
at	23
c_str	24
clear	25
compare(String).....	26
compare(char).....	27
empty	28
find(String).....	29
find(char)	30
find(char,length)	31
find(length).....	32
length	33
operator+=(char)	34
operator+=(character)	35
operator=.....	36
operator==(String)	37
operator==(char)	38
resize.....	39
rfind(String).....	40
rfind(char).....	41
rfind(char,length)	42
rfind(length)	43
size	44
substr	45

sce::Json::Array Class	46
Summary	47
sce::Json::Array	47
Constructors and Destructors	48
Array	48
Array(Array)	49
~Array	50
Public Class Methods	51
back	51
begin	52
clear	53
empty	54
end	55
operator=	56
pop_back	57
push_back	58
size	59
sce::Json::Array::Iterator Class	60
Summary	61
sce::Json::Array::iterator	61
Constructors and Destructors	62
iterator	62
~iterator	63
Public Class Methods	64
advance	64
operator!=	65
operator*	66
operator++	67
operator++ (Postfix)	68
operator->	69
operator=	70
sce::Json::Object Class	71
Summary	72
sce::Json::Object	72
Constructors and Destructors	73
Object	73
Object(Object)	74
~Object	75
Public Class Methods	76
begin	76
clear	77
empty	78
end	79
find	80
insert	81
operator=	82
operator[](String key)	83
size	84

sce::Json::Object::Iterator Class	85
Summary	86
sce::Json::Object::iterator	86
Constructors and Destructors	87
iterator	87
~iterator	88
Public Class Methods	89
advance	89
operator==	90
operator!=	91
operator*	92
operator++	93
operator++ (Postfix)	94
operator->	95
operator=	96
sce::Json::Object::Pair Class	97
Summary	98
sce::Json::Object::Pair	98
Constructors and Destructors	99
Pair	99
Pair(key,Value)	100
~Pair	101
sce::Json::Value Class	102
Summary	103
sce::Json::Value	103
Constructors and Destructors	105
Value	105
Value(Array)	106
Value(Object)	107
Value(String)	108
Value(Value)	109
Value(ValueType)	110
Value(bool)	111
Value(double)	112
Value(int64_t)	113
Value(uint64_t)	114
~Value	115
Type Definition	116
DataReceiveFunction	116
NullAccessFunction	117
Operator Methods	118
operator bool	118
operator=	119
operator[](String key)	120
operator[](char key)	121
operator[](index)	122
Public Instance Methods	123
clear	123

count	124
getArray.....	125
getBoolean.....	126
getInteger.....	127
getObject.....	128
getReal.....	129
getString.....	130
getType.....	131
getUInteger.....	132
getValue(index).....	133
getValue(key).....	134
referArray.....	135
referBoolean.....	136
referInteger.....	137
referObject.....	138
referReal.....	139
referString.....	140
referUInteger.....	141
referValue(index).....	142
referValue(key).....	143
serialize.....	144
serialize(func).....	145
set(Array).....	146
set(Object).....	147
set(String).....	148
set(Value).....	149
set(ValueType).....	150
set(bool).....	151
set(double).....	152
set(int64_t).....	153
set(uint64_t).....	154
setNullAccessCallBack.....	155
swap.....	156
toString.....	157
toString(dst).....	158
sce::Json::Parser Class	159
Summary.....	160
sce::Json::Parser.....	160
Type Definition.....	161
DataProvideFunction.....	161
Public Class Methods.....	162
parse(char).....	162
parse(file).....	163
parse(func).....	164
sce::Json::Initializer Interface Class.....	165
Summary.....	166
sce::Json::Initializer.....	166
Constructors and Destructors	167

Initializer	167
~Initializer	168
Public Instance Methods	169
initialize	169
terminate	170
sce::Json::InitParameter Class	171
Summary	172
sce::Json::InitParameter	172
Constructors and Destructors	173
InitParameter.....	173
InitParameter(param)	174
sce::Json::MemAllocator Interface Class.....	175
Summary	176
sce::Json::MemAllocator	176
Constructors and Destructors	177
MemAllocator	177
~MemAllocator	178
Public Instance Methods	179
allocate	179
deallocate	180
notifyError.....	181

Introduction

000004892117

Library Summary

Library Contents

Item	Description
sce::Json	Json library namespace
sce::Json::String	String class
sce::Json::Array	Array class
sce::Json::Array::iterator	Array::iterator class
sce::Json::Object	Object class
sce::Json::Object::iterator	Object::iterator class
sce::Json::Object::Pair	Object::Pair class
sce::Json::Value	Value class
sce::Json::Parser	Parser class
sce::Json::Initializer	Object initialize interface class
sce::Json::InitParameter	Initialize parameter class
sce::Json::MemAllocator	Memory allocator interface class

sce::Json Namespace

Summary

sce::Json

Json library namespace

Definition

```
namespace Json {}
```

Description

This is the Json library namespace.

Internal classes

Item	Description
sce::Json::String	String class
sce::Json::Array	Array class
sce::Json::Array::iterator	Array::iterator class
sce::Json::Object	Object class
sce::Json::Object::iterator	Object::iterator class
sce::Json::Object::Pair	Object::Pair class
sce::Json::Value	Value class
sce::Json::Parser	Parser class
sce::Json::Initializer	Object initialize interface class
sce::Json::InitParameter	Initialize parameter class
sce::Json::MemAllocator	Memory allocator interface class

Constants

Return Codes

List of return codes returned by the Json library

Definition

Value	(Number)	Description
SCE_JSON_ERROR_PARSE_INVALID_CHAR	0x80920101	Invalid character included in JSON document
SCE_JSON_ERROR_NOMEM	0x80920102	Insufficient memory
SCE_JSON_ERROR_NOFILE	0x80920103	File does not exist
SCE_JSON_ERROR_NOROOT	0x80920104	Value other than a root set for Value
SCE_JSON_ERROR_NOBUF	0x80920105	Buffer is not allocated
SCE_JSON_ERROR_NOINIT	0x80920110	Not initialized
SCE_JSON_ERROR_MULTIPLEINIT	0x80920111	Already initialized

Enumeration Types

ValueType

Value type

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        enum ValueType {
            kValueTypeNull = 0,
            kValueTypeBoolean,
            kValueTypeInteger,
            kValueTypeUInteger,
            kValueTypeReal,
            kValueTypeString,
            kValueTypeArray,
            kValueTypeObject
        };
    }
}
```

Enumeration Values

Value	(Number)	Description
kValueTypeNull	0	NULL
kValueTypeBoolean	1	True/False
kValueTypeInteger	2	Signed integer
kValueTypeUInteger	3	Unsigned integer
kValueTypeReal	4	Floating-point number
kValueTypeString	5	Character string
kValueTypeArray	6	Array
kValueTypeObject	7	Object

Description

These are the enumeration values representing datatypes that can be held in Value objects.

sce::Json::String Class

Summary

sce::Json::String

String class

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class String {};
    }
}
```

Description

This class handles the JSON document string type.
It provides `std::string` subset APIs.

Methods

Method	Description
<code>String()</code>	Constructor
<code>String(String)</code>	Copy constructor
<code>String(char)</code>	Constructor (C language format character array)
<code>~String()</code>	Destructor
<code>append(String)</code>	Append a string to the end of a string
<code>append(char)</code>	Append a string to the end of a string (C language format character array)
<code>append(char, length)</code>	Append a string of the specified length to the end of a string (C language format character array)
<code>at()</code>	Return a character of the specified index
<code>c_str()</code>	Return a character array of the C language format
<code>clear()</code>	Clear a string
<code>compare(String)</code>	Compare two strings
<code>compare(char)</code>	Compare two strings (C language format character array)
<code>empty()</code>	Return true when the string is empty
<code>find(String)</code>	Search for a string from the specified starting position
<code>find(char)</code>	Search for a string from the specified starting position (C language format character array)
<code>find(char, length)</code>	Search for a string for the specified length from the specified starting position (C language format character array)
<code>find(length)</code>	Search for a string for the specified length from the specified starting position
<code>length()</code>	Return the string length
<code>operator+=(char)</code>	Operator to append a string to the end of a string (C language format character array)
<code>operator+=(character)</code>	Operator to append one character to the end of a string
<code>operator=()</code>	Operator to replace the string
<code>operator==(String)</code>	Operator to compare two strings (string type)
<code>operator==(char)</code>	Operator to compare two strings (C language format character array)
<code>resize()</code>	Change the string size

SCE CONFIDENTIAL

Method	Description
<code>rfind(String)</code>	Search for a string from the specified starting position in reversed order
<code>rfind(char)</code>	Search for a string from the specified starting position in reversed order (C language format character array)
<code>rfind(char, length)</code>	Search for a string of a specified length from the specified starting position in reversed order (C language format character array)
<code>rfind(length)</code>	Search for a string of a specified length from the specified starting position in reversed order
<code>size()</code>	Return the string length
<code>substr()</code>	Copy a part of the string as a new string object

000004892117

Constructors and Destructors

String

Constructor

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class String {
            String();
        };
    }
}
```

Return Values

None

Description

This is the default constructor.

String(String)

Copy constructor

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class String {
            String(
                const String& str
            );
        };
    }
}
```

Arguments

str (in) String to provide to the object to be created

Return Values

None

Description

This constructor specifies a string.

The string of the `String` object to be created is a copy of the string specified with *str*.

String(char)

Constructor (C language format character array)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class String {
            String(
                const char* str
            );
        };
    }
}
```

Arguments

str (in) C language format character array to provide to the object to be created

Return Values

None

Description

This constructor specifies a string.

The string of the `String` object is same in content as the C language format character array specified with *str*.

SCE CONFIDENTIAL

~String

Destructor

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class String {
            ~String();
        };
    }
}
```

Return Values

None

Description

This is a destructor.

Public Class Methods

append(String)

Append a string to the end of a string

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class String {
            String& append(
                const String& str
            );
        };
    }
}
```

Arguments

str (in) String to append with

Return Values

Returns the reference of the appended string.

Description

This method appends the specified string to the end of a string.

append(char)

Append a string to the end of a string (C language format character array)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class String {
            String& append(
                const char* str
            );
        };
    }
}
```

Arguments

str (in) C language format character array to append with

Return Values

Returns the reference of the appended string.

Description

This method appends the specified C language format character array to the end of a string.

append(char,length)

Append a string of the specified length to the end of a string (C language format character array)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class String {
            String& append(
                const char* str,
                size_t length
            );
        };
    }
}
```

Arguments

str (in) C language format character array to append with
length (in) Length of string to append

Return Values

Returns the reference of the appended string.

Description

This method appends the specified C language format character array of the length specified to *length* to the end of a string.

at

Return a character of the specified index

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class String {
            unsigned char at(
                size_t pos
            );
        };
    }
}
```

Arguments

pos (in) Index in the string

Return Values

Returns a character code of the specified index.

Returns 0 when the index is outside the valid range.

Description

This method returns one byte of a character code of the specified index.

c_str

Return a character array of the C language format

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class String {
            const char* c_str() const;
        };
    }
}
```

Return Values

Returns a pointer to the C language format character array of the set string.

Description

This method returns a pointer to the C language format character array.

SCE CONFIDENTIAL

clear

Clear a string

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class String {
            void clear();
        };
    }
}
```

Return Values

None

Description

This method clears a string.

compare(String)

Compare two strings

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class String{
            int32_t compare(
                const String& str
            ) const;
        }
    }
}
```

Arguments

str (in) String to compare with

Return Values

Returns zero (0) when the same.

Returns a value other than zero (≠0) when different.

Description

This method compares the string specified in *str* with the current string.

compare(char)

Compare two strings (C language format character array)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class String{
            int32_t compare(
                const char* str
            ) const;
        }
    }
}
```

Arguments

str (in) C language format character array to compare with

Return Values

Returns zero (0) when the same.

Returns a value other than zero (≠0) when different.

Description

This method compares the C language format character array specified in *str* with the current string.

empty

Return true when the string is empty

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class String {
            bool empty() const;
        };
    }
}
```

Return Values

Returns true when string is empty.

Returns false when a string is set.

Description

Returns a boolean value regarding whether or not a string is empty.

find(String)

Search for a string from the specified starting position

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class String {
            size_t find(
                const String& str,
                size_t idx
            );
        };
    }
}
```

Arguments

str (in) String to search for
idx (in) Starting position

Return Values

Returns the position where the string first appears. Returns `String::npos` when not found.

Description

This method searches for the *str* string from the specified starting position and returns the position where the string first appears.

find(char)

Search for a string from the specified starting position (C language format character array)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class String {
            size_t find(
                const char* str,
                size_t idx
            );
        };
    }
}
```

Arguments

str (in) C language format character array to search for
idx (in) Starting position

Return Values

Returns the position where the string first appears. Returns `String::npos` when not found.

Description

This method searches for the *str* C language format character array from the specified starting position and returns the position where the string first appears.

find(char,length)

Search for a string for the specified length from the specified starting position (C language format character array)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class String {
            size_t find(
                const char* str,
                size_t idx,
                size_t siz
            );
        };
    }
}
```

Arguments

str (in) C language format character array to search for
idx (in) Starting position
siz (in) Length of string to search for

Return Values

Returns the position where the string first appears. Returns `String::npos` when not found.

Description

This method searches for the *str* C language format character array for the *siz* length and returns the position where the string first appears.

find(length)

Search for a string for the specified length from the specified starting position

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class String {
            size_t find(
                const String& str,
                size_t idx,
                size_t siz
            );
        };
    }
}
```

Arguments

str (in) String to search for
idx (in) Starting position
siz (in) Length of string to search for

Return Values

Returns the position where the string first appears. Returns `String::npos` when not found.

Description

This method searches for the *str* string for the *siz* length and returns the position where the string first appears.

length

Return the string length

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class String {
            size_t length() const;
        }
    }
}
```

Return Values

Returns the string's number of characters.

Description

This method returns the same value as `size()`.

operator+=(char)

Operator to append a string to the end of a string (C language format character array)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class String {
            String& operator+=(
                const char* str
            );
        };
    }
}
```

Arguments

str (in) C language format character array to append with

Return Values

Returns the reference of the appended string.

Description

This method appends the specified C language format character array to the end of a string.

operator+=(character)

Operator to append one character to the end of a string

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class String {
            String& operator+=(
                unsigned char chr
            );
        };
    }
}
```

Arguments

chr (in) One-character character code to append with

Return Values

Returns the reference of the appended string.

Description

This method appends the specified one-character character code to the end of a string.

operator=

Operator to replace the string

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class String {
            String& operator=(
                const String& str
            );
        };
    }
}
```

Arguments

str (in) String to replace with

Return Values

Returns the reference to the string after replacement.

Description

This method replaces the current string with the specified string.

operator==(String)

Operator to compare two strings (string type)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class String {
            operator==(
                const String& str
            ) const;
        };
    }
}
```

Arguments

str (in) String to compare with

Return Values

Returns true when same as the specified string.

Returns false when different from the specified string.

Description

This method compares the current string with the specified string and returns a boolean value.

operator==(char)

Operator to compare two strings (C language format character array)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class String {
            operator==(
                const char* str
            ) const;
        };
    }
}
```

Arguments

str (in) C language format character array to compare with

Return Values

Returns true when same as the specified C language format character array.

Returns false when different from the specified C language format character array.

Description

This method compares the current string with the specified C language format character array and returns a boolean value.

resize

Change the string size

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class String {
            void resize(
                size_t siz
            );
        };
    }
}
```

Arguments

siz (in) Size after change

Return Values

None

Description

This method changes a string to the specified size.

rfind(String)

Search for a string from the specified starting position in reversed order

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class String {
            size_t rfind(
                const String& str,
                size_t idx
            );
        };
    }
}
```

Arguments

str (in) String to search for
idx (in) Starting position

Return Values

Returns the position where the string last appears. Returns `String::npos` when not found.

Description

This method searches for the *str* string from the specified starting position in reversed order and returns the position where the string last appears.

rfind(char)

Search for a string from the specified starting position in reversed order (C language format character array)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class String {
            size_t rfind(
                const char* str,
                size_t idx
            );
        };
    }
}
```

Arguments

str (in) C language format character array to search for
idx (in) Starting position

Return Values

Returns the position where the string last appears. Returns `String::npos` when not found.

Description

This method searches for the *str* C language format character array from the specified starting position in reversed order and returns the position where the string last appears.

rfind(char,length)

Search for a string of a specified length from the specified starting position in reversed order
(C language format character array)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class String {
            size_t rfind(
                const char* str,
                size_t idx,
                size_t siz
            );
        };
    }
}
```

Arguments

str (in) C language format character array to search for
idx (in) Starting position
siz (in) Length of string to search for

Return Values

Returns the position where the string last appears. Returns `String::npos` when not found.

Description

This method searches for the *str* C language format character array for the *siz* length from the specified starting position in reversed order and returns the position where the string last appears.

rfind(length)

Search for a string for the specified length from the specified starting position in reversed order

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class String {
            size_t rfind(
                const String& str,
                size_t idx,
                size_t siz
            );
        };
    }
}
```

Arguments

str (in) String to search for
idx (in) Starting position
siz (in) Length of string to search for

Return Values

Returns the position where the string last appears. Returns `String::npos` when not found.

Description

This method searches for the *str* string for the *siz* length from the specified starting position in reversed order and returns the position where the string last appears.

size

Return the string length

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class String {
            size_t size() const;
        }
    }
}
```

Return Values

Returns the string's number of characters.

Description

This method returns the same value as `length()`.

substr

Copy a part of the string as a new string object

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class String {
            String substr(
                size_t idx,
                size_t len=npow
            );
        };
    }
}
```

Arguments

idx (in) Starting position with 0 as the base point
len (in) Length (default value: npow)

Return Values

Returns the copied string.

Description

This method copies a string of *len* length from the specified starting position.
When the npow default value is specified to *len*, the string will be copied until the very end.

sce::Json::Array Class

Summary

sce::Json::Array

Array class

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Array {};
    }
}
```

Description

This is a JSON document array type class for handling Value objects as elements. It provides `std::list<Value>` subset APIs.

Methods

Method	Description
<code>Array()</code>	Constructor
<code>Array(Array)</code>	Copy constructor
<code>~Array()</code>	Destructor
<code>back()</code>	Return the last element
<code>begin()</code>	Return iterator pointing to the beginning
<code>clear()</code>	Clear an array
<code>empty()</code>	Return true when array is empty
<code>end()</code>	Return iterator pointing to the end
<code>operator=()</code>	Operator to replace array
<code>pop_back()</code>	Delete the element at the end
<code>push_back()</code>	Add element to the end of the array
<code>size()</code>	Return the number of elements in an array

Constructors and Destructors

Array

Constructor

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Array {
            Array();
        };
    }
}
```

Return Values

None

Description

This is the default constructor.

Array(Array)

Copy constructor

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Array {
            Array(
                const Array& ary
            );
        };
    }
}
```

Arguments

ary (in) Array object to provide to the object to be created

Return Values

None

Description

This constructor specifies an array.

The `Array` object that is created is a copy of the array specified in *ary*.

SCE CONFIDENTIAL

~Array

Destructor

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Array {
            ~Array();
        };
    }
}
```

Return Values

None

Description

This is a destructor.

Public Class Methods

back

Return the last element

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Array {
            Value& back() const;
        }
    }
}
```

Return Values

Returns the reference to the last element.

Description

Returns the reference to the last element.

begin

Return iterator pointing to the beginning

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Array{
            iterator begin() const;
        }
    }
}
```

Return Values

Returns iterator pointing to the beginning.

Description

This method returns an iterator pointing to the beginning.

SCE CONFIDENTIAL

clear

Clear an array

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Array {
            void clear();
        };
    }
}
```

Return Values

None

Description

This method clears an array.

empty

Return true when array is empty

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Array {
            bool empty() const;
        };
    }
}
```

Return Values

Returns true when array is empty.

Returns false when an array is set.

Description

This method returns a boolean value indicating whether or not an array is empty.

end

Return iterator pointing to the end

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Array {
            iterator end() const;
        }
    }
}
```

Return Values

Returns iterator pointing to the end.

Description

This method returns an iterator pointing to the end.

operator=

Operator to replace array

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Array {
            Array& operator=(
                const Array& ary
            );
        };
    }
}
```

Arguments

ary (in) Array to replace with

Return Values

Returns the reference to the array after replacement.

Description

This method replaces an array with the specified array.

SCE CONFIDENTIAL

pop_back

Delete the element at the end

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Array {
            void pop_back();
        };
    }
}
```

Return Values

None

Description

This method deletes the element at the end (Value).

push_back

Add element to the end of the array

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Array {
            void push_back(
                const Value& val
            );
        };
    }
}
```

Arguments

val (in) Element to add

Return Values

None

Description

This method adds an element (*Value*) to the end of the array.

size

Return the number of elements in an array

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Array {
            size_t size() const;
        }
    }
}
```

Return Values

Returns the number of elements in an array.

Description

This method returns the number of elements in an array.

sce::Json::Array::Iterator Class

Summary

sce::Json::Array::iterator

Array::iterator class

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Array {
            class iterator{
            };
        };
    }
}
```

Description

This is an input iterator of the Array class.

Methods

Method	Description
iterator()	Constructor
~iterator()	Destructor
advance()	Increment pointed position by the offset amount
operator!=()	Operator to return true when pointed position differs
operator*()	Operator to return reference to an element
operator++()	Operator to increment pointed position
operator++ (postfix)	Operator to increment pointed position (postfix)
operator->()	Operator to return a pointer to an element
operator=()	Operator to replace the iterator

Constructors and Destructors

iterator

Constructor

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Array {
            class iterator{
                iterator();
            }
        };
    }
}
```

Return Values

None

Description

This is the default constructor.

SCE CONFIDENTIAL

~iterator

Destructor

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Array {
            class iterator{
                ~iterator();
            }
        };
    }
}
```

Return Values

None

Description

This is a destructor.

Public Class Methods

advance

Increment pointed position by the offset amount

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Array {
            class iterator{
                void advance(
                    size_t offset
                );
            };
        };
    };
}
```

Arguments

offset (in) Offset

Return Values

None

Description

This method increments the pointed position by the specified offset amount.

operator!=

Operator to return true when pointed position differs

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Array {
            class iterator{
                bool operator!=(
                    iterator it
                ) const;
            };
        };
    };
}
```

Arguments

it (in) Iterator to compare with

Return Values

Returns true when the pointed position differs with the specified iterator.
Returns false when the pointed position is the same as the specified iterator.

Description

This method compares the pointed position and returns a boolean value.

operator*

Operator to return reference to an element

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Array {
            class iterator{
                Value& operator*() const;
            };
        };
    };
}
```

Return Values

Returns the reference to the element corresponding to the pointed position.

Description

This method returns the reference to the element at the pointed position.

operator++

Operator to increment pointed position

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Array {
            class iterator{
                iterator& operator++();
            };
        };
    };
}
```

Return Values

Returns the reference to the iterator after it is moved.

Description

This method increments the pointed position.

operator++ (Postfix)

Operator to increment pointed position (postfix)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Array {
            class iterator{
                iterator& operator++(int);
            }
        };
    }
}
```

Return Values

Returns the reference to the iterator after it is moved.

Description

This method increments the pointed position.

operator->

Operator to return a pointer to an element

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Array {
            class iterator{
                Value* operator->() const;
            };
        };
    };
}
```

Return Values

Returns a pointer to the element corresponding to the pointed position.

Description

This method returns a pointer to the element at the pointed position.

operator=

Operator to replace the iterator

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Array {
            class iterator{
                iterator& operator=(
                    const iterator& it
                );
            };
        };
    };
}
```

Arguments

it (in) Iterator to replace with

Return Values

Returns the reference to the iterator after replacement.

Description

This method replaces an iterator with the specified iterator.

sce::Json::Object Class

000004892117

Summary

sce::Json::Object

Object class

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Object {};
    }
}
```

Description

This is a JSON document object type class for handling `sce::Json::Object::Pair` as elements. It provides `std::map<String, Value>` subset APIs.

Methods

Method	Description
<code>Object()</code>	Constructor
<code>Object(Object)</code>	Copy constructor
<code>~Object()</code>	Destructor
<code>begin()</code>	Return iterator pointing to the beginning
<code>clear()</code>	Clear a JSON document object
<code>empty()</code>	Return true when the element is empty
<code>end()</code>	Return iterator pointing to the end
<code>find()</code>	Return iterator pointing to the element corresponding to the specified key
<code>insert()</code>	Add an element
<code>operator=()</code>	Operator to replace a JSON document object
<code>operator[] (String key)</code>	Operator to return reference to the <code>Value</code> object
<code>size()</code>	Return the number of elements in the JSON document object

Constructors and Destructors

Object

Constructor

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Object {
            Object();
        };
    }
}
```

Return Values

None

Description

This is the default constructor.

Object(Object)

Copy constructor

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Object {
            Object(
                const Object& obj
            );
        };
    }
}
```

Arguments

obj (in) Reference of a JSON document object to provide to the object to be created

Return Values

None

Description

This constructor specifies a JSON document object.

The object to be created will be a copy of the JSON document object specified with *obj*.

SCE CONFIDENTIAL

~Object

Destructor

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Object {
            ~Object();
        };
    }
}
```

Return Values

None

Description

This is a destructor.

Public Class Methods

begin

Return iterator pointing to the beginning

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Object{
            iterator begin() const;
        }
    }
}
```

Return Values

Returns iterator pointing to the beginning.

Description

This method returns an iterator pointing to the beginning.

clear

Clear a JSON document object

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Object {
            void clear();
        };
    }
}
```

Return Values

None

Description

This method clears a JSON document object.

empty

Return true when the element is empty

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Object {
            bool empty() const;
        };
    }
}
```

Return Values

Returns true when element is empty.

Returns false when an element is set.

Description

This method returns whether or not an element is empty as a boolean value.

end

Return iterator pointing to the end

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Object {
            iterator end() const;
        }
    }
}
```

Return Values

Returns iterator pointing to the end.

Description

This method returns an iterator pointing to the end.

find

Return iterator pointing to the element corresponding to the specified key

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Object {
            iterator find(
                const String& key
            ) const;
        }
    }
}
```

Arguments

key (in) Search key string

Return Values

Returns the corresponding iterator.

Description

This method returns an iterator pointing to the element corresponding to the specified key string.

insert

Add an element

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Array {
            void insert(
                const Pair& objpair
            );
        };
    }
}
```

Arguments

objpair (in) Element to add

Return Values

None

Description

This method adds the specified element (`sce::Json::Object::Pair`).
The element will not be inserted if the key value specified with
`sce::Json::Object::Pair::first` is redundant.

operator=

Operator to replace a JSON document object

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Object {
            Object& operator=(
                const Object& obj
            );
        };
    }
}
```

Arguments

obj (in) JSON document object to replace with

Return Values

Returns the reference to the JSON document object after replacement.

Description

This method replaces a JSON document object with the specified JSON document object.

operator[](String key)

Operator to return reference to the Value object

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Object {
            Value& operator[] (
                const String& key
            );
        };
    }
}
```

Arguments

key (in) Key string

Return Values

Returns reference to the Value object corresponding to the key string.

Description

Of the object-type elements, this operator accesses the Value object with a value that corresponds to the key indicated by the character string specified in *key*.

size

Return the number of elements in the JSON document object

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Object {
            size_t size() const;
        }
    }
}
```

Return Values

Returns the number of elements in a JSON document object.

Description

This method returns the number of elements in a JSON document object.

sce::Json::Object::Iterator Class

Summary

sce::Json::Object::iterator

Object::iterator class

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Object {
            class iterator{
            };
        };
    }
}
```

Description

This is the input iterator of the `Object` class.

Methods

Method	Description
<code>iterator()</code>	Constructor
<code>~iterator()</code>	Destructor
<code>advance()</code>	Increment pointed position by the offset amount
<code>operator==()</code>	Operator to return true when pointed position is the same
<code>operator!=()</code>	Operator to return true when pointed position differs
<code>operator*()</code>	Operator to return reference to an element
<code>operator++()</code>	Operator to increment pointed position
<code>operator++ (postfix)</code>	Operator to increment pointed position (postfix)
<code>operator->()</code>	Operator to return a pointer to an element
<code>operator=()</code>	Operator to replace the iterator

Constructors and Destructors

iterator

Constructor

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Object {
            class iterator{
                iterator();
            }
        };
    }
}
```

Return Values

None

Description

This is the default constructor.

SCE CONFIDENTIAL

~iterator

Destructor

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Object{
            class iterator{
                ~iterator();
            }
        };
    }
}
```

Return Values

None

Description

This is a destructor.

Public Class Methods

advance

Increment pointed position by the offset amount

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Object{
            class iterator{
                void advance(
                    size_t offset
                );
            };
        };
    };
}
```

Arguments

offset (in) Offset

Return Values

None

Description

This method increments the pointed position by the specified offset amount.

operator==

Operator to return true when pointed position is the same

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Object{
            class iterator{
                bool operator==(
                    iterator it
                ) const;
            };
        };
    };
}
```

Arguments

it (in) Iterator to compare with

Return Values

Returns true when the pointed position is the same as the specified iterator.

Returns false when the pointed position differs with the specified iterator.

Description

This method compares the pointed position and returns a boolean value.

operator!=

Operator to return true when pointed position differs

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Object{
            class iterator{
                bool operator!=(
                    iterator it
                ) const;
            }
        };
    }
}
```

Arguments

it (in) Iterator to compare with

Return Values

Returns true when the pointed position differs with the specified iterator.
Returns false when the pointed position is the same as the specified iterator.

Description

This method compares the pointed position and returns a boolean value.

operator*

Operator to return reference to an element

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Object{
            class iterator{
                Pair& operator*() const;
            }
        };
    }
}
```

Return Values

Returns the reference to the element corresponding to the pointed position.

Description

This method returns the reference to the element at the pointed position.

operator++

Operator to increment pointed position

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Object {
            class iterator{
                iterator& operator++();
            }
        };
    }
}
```

Return Values

Returns the reference to the iterator after it is moved.

Description

This method increments the pointed position.

operator++ (Postfix)

Operator to increment pointed position (postfix)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Object{
            class iterator{
                iterator& operator++(int);
            }
        };
    }
}
```

Return Values

Returns the reference to the iterator after it is moved.

Description

This method increments the pointed position.

operator->

Operator to return a pointer to an element

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Object{
            class iterator{
                Pair* operator->() const;
            }
        };
    }
}
```

Return Values

Returns a pointer to the element corresponding to the pointed position.

Description

This method returns a pointer to the element at the pointed position.

operator=

Operator to replace the iterator

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Object{
            class iterator{
                iterator& operator=(
                    const iterator& it
                );
            };
        };
    };
}
```

Arguments

it (in) Iterator to replace with

Return Values

Returns the reference to the iterator after replacement.

Description

This method replaces an iterator with the specified iterator.

sce::Json::Object::Pair Class

Summary

sce::Json::Object::Pair

Object::Pair class

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Object {
            class Pair{
            };
        };
    }
}
```

Description

This is a class for handling elements held by the `Object` class; this class holds a combination of a key (`String`) and value (`Value`) as members.

Members

Variable	Type	Description
<i>first</i>	String	String to be the element key
<i>second</i>	Value	Element value

Methods

Method	Description
<code>Pair()</code>	Constructor
<code>Pair(key, Value)</code>	Constructor (keystring, value)
<code>~Pair()</code>	Destructor

Constructors and Destructors

Pair

Constructor

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Object {
            class Pair{
                Pair();
            }
        };
    }
}
```

Return Values

None

Description

This is the default constructor.

Pair(key,Value)

Constructor (key string, value)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Object {
            class Pair{
                Pair(
                    const String& key,
                    const Value& val
                );
            };
        };
    };
}
```

Arguments

key (in) Key string
val (in) Set value

Return Values

None

Description

This method creates `Pair` with the specified value.

SCE CONFIDENTIAL

~Pair

Destructor

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Object {
            class Pair{
                ~Pair();
            }
        };
    }
}
```

Return Values

None

Description

This is a destructor.

sce::Json::Value Class

000004892117

Summary

sce::Json::Value

Value class

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {};
    }
}
```

Description

This class represents elements in a JSON document. It maintains element types (ValueType) associated one to one with JSON document elements, and values.

When a JSON document is parsed, a Value object tree can be obtained. In addition, by setting a value for a Value object and formulating a tree structure for serialization, a JSON document can be created.

Methods

Method	Description
Value()	Constructor
Value(Array)	Constructor (array)
Value(Object)	Constructor (object)
Value(String)	Constructor (character string)
Value(Value)	Copy constructor
Value(ValueType)	Constructor (type specification)
Value(bool)	Constructor (truth value)
Value(double)	Constructor (floating-point number)
Value(int64_t)	Constructor (signed integer)
Value(uint64_t)	Constructor (unsigned integer)
~Value()	Destructor
operator bool()	Truth value conversion
operator=()	Assignment operator
operator[] (String key)	Subscript operator (String type key character string)
operator[] (char key)	Subscript operator (char type key character string)
operator[] (index)	Subscript operator (index)
clear()	Initialize Value object
count()	Get element count
getArray()	Get reference to const array
getBoolean()	Get reference to const truth value
getInteger()	Get reference to const signed integer
getObject()	Get reference to const object
getReal()	Get reference to const floating-point number
getString()	Get reference to const character string
getType()	Get ValueType
getUInteger()	Get reference to const unsigned integer
getValue(index)	Get reference to const Value (index)

Method	Description
getValue (key)	Get reference to const Value (key character string)
referArray ()	Get pointer to array
referBoolean ()	Get pointer to truth value
referInteger ()	Get pointer to signed integer
referObject ()	Get pointer to object
referReal ()	Get pointer to floating-point number
referString ()	Get pointer to character string
referUInteger ()	Get pointer to unsigned integer
referValue (index)	Get pointer to Value object (index)
referValue (key)	Get pointer to Value object (key character string)
serialize ()	Serialize
serialize (func)	Partition and serialize
set (Array)	Set value (array)
set (Object)	Set value (object)
set (String)	Set value (character string)
set (Value)	Set value (copy)
set (ValueType)	Set ValueType
set (bool)	Set value (truth value)
set (double)	Set value (floating-point number)
set (int64_t)	Set value (signed integer)
set (uint64_t)	Set value (unsigned integer)
setNullAccessCallBack ()	Set NULL access callback
swap ()	Swap data
toString ()	Convert value to character string
toString (dst)	Convert value to character string (parameter pass)

Constructors and Destructors

Value

Constructor

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            Value();
        };
    }
}
```

Return Values

None

Description

This is the default constructor.

The ValueType of the generated Value object is kValueTypeNull and the value is 0 (NULL).

Value(Array)

Constructor (array)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            Value(
                const Array& a
            );
        };
    }
}
```

Arguments

a (in) value assigned to the object to generate

Return Values

None

Description

This is a constructor that specifies an array.

The ValueType of the generated Value object is kValueTypeArray and the value is a copy of the array specified with *a*.

Value(Object)

Constructor (object)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            Value(
                const Object& o
            );
        };
    }
}
```

Arguments

- (in) value assigned to the object to generate

Return Values

None

Description

This is a constructor that specifies an object.

The ValueType of the generated Value object is kValueTypeObject and the value is a copy of the object specified with *o*.

Value(String)

Constructor (character string)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            Value(
                const String& s
            );
        };
    }
}
```

Arguments

s (in) value assigned to the object to generate

Return Values

None

Description

This is a constructor that specifies a character string.

The ValueType of the generated Value object is kValueTypeString and the value is a copy of the character string specified with *s*.

SCE CONFIDENTIAL

Value(Value)

Copy constructor

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            Value(
                const Value& x
            );
        };
    }
}
```

Arguments

x (in) Value value that will be the copy source

Return Values

None

Description

This is a copy constructor that copies the Value value.

Value(ValueType)

Constructor (type specification)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            Value(
                ValueType type
            );
        };
    }
}
```

Arguments

type (in) ValueType

Return Values

None

Description

This is a constructor that specifies a ValueType.

The ValueType of the generated Value object is *type* and the value is the initial value of each of the following types shown in the following.

- kValueTypeNull: 0(NULL)
- kValueTypeBoolean: false
- kValueTypeInteger: 0
- kValueTypeUInteger: 0
- kValueTypeReal: 0
- kValueTypeString: character string with a length of 0
- kValueTypeArray: array with 0 elements
- kValueTypeObject: object with 0 elements

Value(bool)

Constructor (truth value)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            Value(
                bool b
            );
        };
    }
}
```

Arguments

b (in) value assigned to the object to generate

Return Values

None

Description

This is a constructor that specifies a truth value.

The ValueType of the generated Value object is kValueTypeBoolean and the value is the value specified with *b*.

Value(double)

Constructor (floating-point number)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            Value(
                double n
            );
        };
    }
}
```

Arguments

n (in) value assigned to the object to generate

Return Values

None

Description

This is a constructor that specifies a floating-point number.

The ValueType of the generated Value object is kValueTypeReal and the value is the value specified with *n*.

Value(int64_t)

Constructor (signed integer)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            Value(
                int64_t l
            );
        };
    }
}
```

Arguments

l (in) value assigned to the object to generate

Return Values

None

Description

This is a constructor that specifies a signed integer.

The ValueType of the generated Value object is `kValueTypeInteger` and the value is the value specified with *l*.

Value(uint64_t)

Constructor (unsigned integer)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            Value(
                uint64_t ul
            );
        };
    }
}
```

Arguments

ul (in) value assigned to the object to generate

Return Values

None

Description

This is a constructor that specifies an unsigned integer.

The ValueType of the generated Value object is `kValueTypeUInteger` and the value is the value specified with *ul*.

SCE CONFIDENTIAL

~Value

Destructor

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            ~Value();
        };
    }
}
```

Return Values

None

Description

This is a destructor.

Type Definition

DataReceiveFunction

Data receive callback function

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            typedef int32_t (*DataReceiveFunction) (
                String& buf,
                void* userdata
            );
        };
    };
}
```

Arguments

buf (in) Part of the serialization results
userdata (in) User-specified information specified with `serialize()`

Return Values

Return 0 if the processing terminates normally.
 Return a negative value if you want to interrupt the serialization processing due to abnormalities, etc.

Description

This is an interface definition for the callback function that partitions and receives the serialization results. When a function that follows this specification is implemented and passed as an argument to `serialize()`, it will be called back every time a single element is serialized.

To *buf*, a reference to the character string that represents part of the serialized results will be passed.
 To *userdata*, the value specified for the argument *userdata* of `serialize()` will be passed as-is.
 Based on this information, perform appropriate processing such as outputting the serialization results to a file or sending them to a network.

Return a negative value if you want to interrupt the serialization processing. In such cases, `serialize()` will return the value returned by the callback as-is.

See Also

`serialize(func)`

NullAccessFunction

NULL access callback function

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            typedef const Value& (*NullAccessFunction) (
                ValueType accesstype,
                const Value* parent,
                void* context
            );
        };
    }
}
```

Arguments

accesstype (in) ValueType for which access was attempted
parent (in) Parent Value object
context (in) Information specified upon callback function setting

Return Values

Return a reference to the Value object for the access result.

Description

This is an interface definition for the callback function that customizes the behavior for NULL. When a function that follows this specification is implemented and set for the NULL access callback using `setNullAccessCallBack()`, it will be called back when a value get method (`getXXX()`) is called for a Value object with a ValueType of `kValueTypeNull`.

To *accesstype*, the ValueType that represents which value get method was called will be passed. For example, `kValueTypeString` will be passed if `getString()` was called.

To *parent*, a pointer to the Value object that is the parent of the accessed Value object will be passed.

To *context*, the value specified for the argument *context* of `setNullAccessCallBack()` when the NULL access callback was set will be passed as-is.

Determine the values that the called value get methods should return based on this information, and return them as references to the Value objects. In addition to returning values, it is also possible to cause errors.

See Also

`setNullAccessCallBack()`

Operator Methods

operator bool

Truth value conversion

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            operator bool() const;
        }
    }
}
```

Return Values

Returns the truth value.

Description

Returns truth values depending on the ValueType as in the following.

- kValueTypeNull: false
- kValueTypeBoolean: Setting value
- kValueTypeInteger: false when 0, true when non-0
- kValueTypeUInteger: false when 0, true when non-0
- kValueTypeReal: false when 0, true when non-0
- kValueTypeString: false when a length of 0, true when a length of 1 or greater
- kValueTypeArray: true
- kValueTypeObject: true

operator=

Assignment operator

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            Value& operator=(
                const Value& x
            );
        };
    };
}
```

Arguments

x (in) Copy source Value object

Return Values

Returns a reference to the copied Value object.

Description

Returns a copy of the Value object specified with x .

operator[](String key)

Subscript operator (String type key character string)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            const Value& operator[] (
                const String& key
            ) const;
        }
    }
}
```

Arguments

key (in) Key character string

Return Values

Returns a reference to the Value for the key character string.

Description

This is an operator that accesses a character string specified with *key* from among the object type Value object elements as a key.

When this operation is performed for a Value object other than an object type, or if a character string other than a key that an object maintains is specified, a reference to a Value object with `kValueTypeNull` specified for the Value Type will be returned.

operator[](char key)

Subscript operator (char type key character string)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            const Value& operator[] (
                const char* key
            ) const;
        }
    }
}
```

Arguments

key (in) key character string

Return Values

Returns a reference to a `Value` object compatible with key character string.

Description

This is an operator that accesses a character string specified with *key* from among the object type `Value` object elements as a key.

When this operation is performed for a `Value` object other than an object type, or if a character string other than a key that an object maintains is specified, a reference to a `Value` object with `kValueTypeNull` specified for the `Value` Type will be returned.

operator[](index)

Subscript operator (index)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            const Value& operator[] (
                size_t index
            ) const;
        }
    }
}
```

Arguments

index (in) Index value

Return Values

Returns a reference to the `Value` object for *index*.

Description

This is an operator that accesses elements of an array or object type `Value` object based on the subscript (*index*).

When operation is performed for a `Value` object other than an array or object, a reference to a `Value` object with `kValueTypeNull` set for the `ValueType` will be returned. This will be the same in cases where a value that exceeds the array length or number of object elements is specified for *index*.

Public Instance Methods

clear

Initialize Value object

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            void clear();
        }
    }
}
```

Return Values

None

Description

Specify `kValueTypeNull` for the `ValueType` of this `Value` object and specify 0 (NULL) for the value.

SCE CONFIDENTIAL

count

Get element count

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            int32_t count() const;
        }
    }
}
```

Return Values

Returns the element count.

Description

Returns the array or object element count.

When this operation is performed for a Value object with a ValueType that is not kValueTypeArray or kValueTypeObject, 0 will return.

getArray

Get reference to cons array

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            const Array& getArray() const;
        }
    }
}
```

Return Values

Returns a reference to the const array.

Description

Gets the array set for this Value object.

If the ValueType is `kValueTypeNull` and a NULL access callback function is set, a reference to the Value object value returned by the callback function will be returned.

If the ValueType is `kValueTypeNull` and a NULL access callback function is not set, or if the ValueType is not `kValueTypeArray` or `kValueTypeNull`, a reference to an array with 0 elements will be returned.

getBoolean

Get reference to const truth value

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            const bool& getBoolean() const;
        }
    }
}
```

Return Values

Returns a reference to the const truth value.

Description

Gets the truth value set for this Value object.

If the ValueType is kValueTypeNull and a NULL access callback function is set, a reference to the Value object value returned by the callback function will be returned. If the ValueType is kValueTypeNull and a NULL access callback function is not set, or if the ValueType is not kValueTypeBoolean or kValueTypeNull, a reference to a truth value with a value of false will be returned.

getInteger

Get reference to const signed integer

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            const int64_t& getInteger() const;
        }
    }
}
```

Return Values

Returns a reference to the const signed integer.

Description

Gets the signed integer set for this Value object.

If the ValueType is kValueTypeNull and a NULL access callback function is set, a reference to the Value object value returned by the callback function will be returned.

If the ValueType is kValueTypeNull and a NULL access callback function is not set, or if the ValueType is not kValueTypeInteger, kValueTypeUInteger, or kValueTypeNull, a reference to a signed integer with a value of 0 will be returned.

getObject

Get reference to const object

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            const Object& getObject() const;
        }
    }
}
```

Return Values

Returns a reference to the const object.

Description

Gets the object set for this Value object.

If the ValueType is kValueTypeNull and a NULL access callback function is set, a reference to the Value object value returned by the callback function will be returned.

If the ValueType is kValueTypeNull and a NULL access callback function is not set, or if the ValueType is not kValueTypeObject or kValueTypeNull, a reference to an object with 0 elements will be returned.

getReal

Get reference to const floating-point number

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            const double& getReal() const;
        }
    }
}
```

Return Values

Returns a reference to the const floating-point number.

Description

Gets the floating-point number set for this Value object.

If the ValueType is kValueTypeNull and a NULL access callback function is set, a reference to the Value object value returned by the callback function will be returned. If the ValueType is kValueTypeNull and a NULL access callback function is not set, or if the ValueType is not kValueTypeReal or kValueTypeNull, a reference to a floating-point number with a value of 0.0f will be returned.

getString

Get reference to const character string

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            const String& getString() const;
        }
    }
}
```

Return Values

Returns a reference to the const character string.

Description

Gets the character string set for this Value object.

If the ValueType is kValueTypeNull and a NULL access callback is set, a reference to the Value object value returned by the callback function will be returned.

If the ValueType is kValueTypeNull and a NULL access callback is not set, or if the ValueType is not kValueTypeString or kValueTypeNull, a reference to a character string with a length of 0 will be returned.

SCE CONFIDENTIAL

getType

Get ValueType

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            ValueType getType() const;
        }
    }
}
```

Return Values

Returns the set ValueType.

Description

Gets the ValueType set for this Value object.

getUInteger

Get reference to const unsigned integer

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            const uint64_t& getUInteger() const;
        }
    }
}
```

Return Values

Returns a reference to the const unsigned integer.

Description

Gets the unsigned integer set for this Value object.

If the ValueType is kValueTypeNull and a NULL access callback function is set, a reference to the Value object value returned by the callback function will be returned. If the ValueType is kValueTypeNull and a NULL access callback function is not set, or if the ValueType is not kValueTypeUInteger, kValueTypeInteger, or kValueTypeNull, a reference to an unsigned integer with a value of 0 will be returned.

getValue(index)

Get reference to const Value (index)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            const Value& getValue(
                size_t index
            ) const;
        }
    }
}
```

Arguments

index (in) index

Return Values

Returns a reference to the const Value.

Description

Returns a reference to the one that is applicable to *index* from among the array or object type Value object elements.

When this operation is performed for a Value object with a ValueType that is not `kValueTypeArray` or `kValueTypeObject`, a reference to a Value object with `kValueTypeNull` set for the ValueType will be returned.

getValue(key)

Get reference to const Value (key character string)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            const Value& getValue(
                const String& key
            ) const;
        }
    }
}
```

Arguments

key (in) Key character string

Return Values

Returns a reference to the const Value.

Description

Returns a reference to the character string specified with *key* that will be the key from among the object type Value objects.

When this operation is performed for a Value object with a ValueType that is not `kValueTypeObject`, a reference to a Value object with `kValueTypeNull` set for the ValueType will be returned.

referArray

Get pointer to array

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            Array* referArray();
        }
    }
}
```

Return Values

Returns a pointer to the array.

Description

Gets the array set for this Value object.

Returns NULL if the ValueType is not kValueTypeArray.

Notes

The return value pointer points to the actual array set for the Value object. It is possible to directly edit the object value through the returned pointer, but note that access will not be possible if the Value object is deleted. In addition, release processing (delete, etc.) must not be performed using this pointer.

referBoolean

Get pointer to truth value

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            bool* referBoolean();
        }
    }
}
```

Return Values

Returns a pointer to the truth value.

Description

Gets the truth value set for this Value object.

Returns NULL if the ValueType is not kValueTypeBoolean.

Notes

The return value pointer points to the actual truth value set for the Value object. It is possible to directly edit the object value through the returned pointer, but note that access will not be possible if the Value object is deleted. In addition, release processing (delete, etc.) must not be performed using this pointer.

referInteger

Get pointer to signed integer

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            int64_t* referInteger();
        }
    }
}
```

Return Values

Returns a pointer to the signed integer.

Description

Gets the signed integer set for this Value object.

Returns NULL if the ValueType is not kValueTypeInteger or kValueTypeUInteger.

Notes

The return value pointer points to the actual signed integer set for the Value object. It is possible to directly edit the object value through the returned pointer, but note that access will not be possible if the Value object is deleted. In addition, release processing (delete, etc.) must not be performed using this pointer.

referObject

Get pointer to object

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            Object* referObject();
        }
    }
}
```

Return Values

Returns a pointer to the object.

Description

Gets the object set for this Value object.

Returns NULL if the ValueType is not kValueTypeObject.

Notes

The return value pointer points to the actual object set for the Value object. It is possible to directly edit the object value through the returned pointer, but note that access will not be possible if the Value object is deleted. In addition, release processing (delete, etc.) must not be performed using this pointer.

referReal

Get pointer to floating-point number

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            double* referReal();
        }
    }
}
```

Return Values

Returns a pointer to the floating-point number.

Description

Gets the floating-point number set for this Value object.
Returns NULL if the ValueType is not kValueTypeReal.

Notes

The return value pointer points to the actual floating-point number set for the Value object. It is possible to directly edit the object value through the returned pointer, but note that access will not be possible if the Value object is deleted. In addition, release processing (delete, etc.) must not be performed using this pointer.

referString

Get pointer to character string

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            String* referString();
        }
    }
}
```

Return Values

Return a pointer to the character string.

Description

Gets the character string set for this Value object.

If the ValueType is not kValueTypeString, NULL will be returned.

Notes

The return value pointer points to the actual character string set for the Value object. It is possible to directly edit the object value through the returned pointer, but note that access will not be possible if the Value object is deleted. In addition, release processing (delete, etc.) must not be performed using this pointer.

referUInteger

Get pointer to unsigned integer

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            uint64_t* referUInteger();
        }
    }
}
```

Return Values

Returns a pointer to the unsigned integer.

Description

Gets the unsigned integer set for this Value object.

Returns NULL if the ValueType is not kValueTypeUInteger or kValueTypeInteger.

Notes

The return value pointer points to the actual unsigned integer set for the Value object. It is possible to directly edit the object value through the returned pointer, but note that access will not be possible if the Value object is deleted. In addition, release processing (delete, etc.) must not be performed using this pointer.

SCEI CONFIDENTIAL

referValue(index)

Get pointer to Value object (index)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            Value* referValue(
                size_t index
            );
        };
    };
}
```

Arguments

index (in) index

Return Values

Returns a pointer to the child Value object applicable to *index*.

Description

Returns a pointer to the one that is applicable to *index* from among the children of this Value object. Returns NULL if the ValueType is not `kValueTypeArray` or `kValueTypeObject`.

Notes

The return value pointer points to the actual child Value object. It is possible to directly edit the object value through the returned pointer, but note that access will not be possible if the Value object is deleted. In addition, release processing (delete, etc.) must not be performed using this pointer.

referValue(key)

Get pointer to Value object (key character string)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            Value* referValue(
                const String& key
            );
        };
    };
}
```

Arguments

key (in) Key character string

Return Values

Returns a pointer to the child Value object applicable to the key character string.

Description

Returns a pointer to the character string specified with *key* that will be the key.
Returns NULL if the ValueType is not kValueTypeObject.

Notes

The return value pointer points to the actual child value set for the Value object. It is possible to directly edit the object value through the returned pointer, but note that access will not be possible if the Value object is deleted. In addition, release processing (delete, etc.) must not be performed using this pointer.

serialize

Serialize

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            int32_t serialize(
                String& dst
            );
        }
    }
}
```

Arguments

dst (out) Reference to the character string for the serialization results

Return Values

Returns SCE_OK (=0) for normal termination.

Returns an error code (negative value) for errors (refer to the "[Return Codes](#)" section for details).

Description

Serializes this Value object and the subordinate Value objects, then generates a JSON document.

serialize(func)

Partition and serialize

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            int32_t serialize(
                String& buf,
                DataReceiveFunction func,
                void *userdata
            );
        };
    };
}
```

Arguments

buf (in) Buffer to store the character string for the serialization results
func (in) Data receive callback function
userdata (in) User-defined information to pass to the data receive callback function

Return Values

Returns SCE_OK (=0) for normal termination.

Returns an error code (negative value) for errors (refer to the "[Return Codes](#)" section for details).

Description

Serializes this Value object and the subordinate Value objects, then generates a JSON document. The data receive callback function specified with *func* will be called every time a single Value object is serialized, and part of the generated JSON document will be passed every time. The callback function will also be called upon element start and stop in regards to arrays and objects.

set(Array)

Set value (array)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            void set(
                const Array& a
            );
        }
    }
}
```

Arguments

a (in) Array to set

Return Values

None

Description

Sets a copy of *a* for the value of this `Value` object and sets `kValueTypeArray` for the `ValueType`.

set(Object)

Set value (object)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            void set(
                const Object& o
            );
        }
    }
}
```

Arguments

- (in) Object to set

Return Values

None

Description

Sets a copy of *o* for the value of this *Value* object and sets *kValueTypeObject* for the *ValueType*.

set(String)

Set value (character string)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            void set(
                const String& s
            );
        }
    }
}
```

Arguments

s (in) Character string to set

Return Values

None

Description

Sets a copy of *s* for the value of this Value object and sets `kValueTypeString` for the `ValueType`.

SCE CONFIDENTIAL

set(Value)

Set value (copy)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            void set(
                const Value& x
            );
        }
    }
}
```

Arguments

x (in) Copy source Value object

Return Values

None

Description

Copies the Value object specified with x to this Value object.

set(ValueType)

Set ValueType

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            void set(
                ValueType type
            );
        }
    }
}
```

Arguments

type (in) ValueType to set

Return Values

None

Description

Sets *type* for the ValueType of this Value object and initializes the value according to type as in the following.

- kValueTypeNull: 0 (NULL)
- kValueTypeBoolean: false
- kValueTypeInteger: 0
- kValueTypeUInteger: 0
- kValueTypeReal: 0
- kValueTypeString: character string with a length of 0
- kValueTypeArray: array with 0 elements
- kValueTypeObject: object with 0 elements

set(bool)

Set value (truth value)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            void set(
                bool b
            );
        }
    }
}
```

Arguments

b (in) Truth value to set

Return Values

None

Description

Sets *b* for the value of this Value object and sets `kValueTypeBoolean` for the `ValueType`.

set(double)

Set value (floating-point number)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            void set(
                double n
            );
        }
    }
}
```

Arguments

n (in) Floating-point number to set

Return Values

None

Description

Sets *n* for the value of this `Value` object and sets `kValueTypeReal` for the `ValueType`.

SCE CONFIDENTIAL

set(int64_t)

Set value (signed integer)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            void set(
                int64_t l
            );
        }
    }
}
```

Arguments

l (in) Signed integer to set

Return Values

None

Description

Sets *l* for the value of this Value object and sets `kValueTypeInteger` for the `ValueType`.

SCE CONFIDENTIAL

set(uint64_t)

Set value (unsigned integer)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            void set(
                uint64_t ul
            );
        }
    }
}
```

Arguments

ul (in) Unsigned integer to set

Return Values

None

Description

Sets *ul* for the value of this Value object and sets `kValueTypeUInteger` for the `ValueType`.

setNullAccessCallback

Set NULL access callback

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            int32_t setNullAccessCallback(
                NullAccessFunction func,
                void* context
            );
        };
    };
}
```

Arguments

func (in) NULL access callback function
context (in) Arbitrary information to pass to the NULL access callback function

Return Values

Returns SCE_OK (=0) for normal termination.

Returns an error code (negative value) for errors (refer to the "[Return Codes](#)" section for details).

Description

This function sets the NULL access callback function.

When a value get method (`getXXX()`) is called for a `Value` object with a `ValueType` that is `kValueTypeNull`, the callback function specified with *func* will be called.

SCE CONFIDENTIAL

swap

Swap data

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            void swap(
                Value& rhs
            );
        };
    }
}
```

Arguments

rhs (in/out) Value object to convert

Return Values

None

Description

Swaps the ValueType and value between this Value object and the Value object specified with *rhs*.

toString

Convert value to character string

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            inline const String toString() const;
        }
    }
}
```

Return Values

Returns the converted character string.

Description

Converts the value set for this Value object to a character string.

The conversion results will be returned as follows according to ValueType.

- kValueTypeInteger, kValueTypeUInteger, kValueTypeReal: Returns a digit sequence that represents the set value.
- kValueTypeString: Returns the set character string as-is.
- kValueTypeBoolean: Returns "false" or "true" depending on the set truth value.
- kValueTypeArray: Returns "array".
- kValueTypeObject: Returns "object".
- kValueTypeNull: Returns "null".

toString(dst)

Convert value to character string (parameter pass)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Value {
            void toString(
                String& dst
            ) const;
        }
    }
}
```

Arguments

dst (out) Reference to the converted character string

Return Values

None

Description

Converts the value set for this Value object to a character string.

The conversion results are returned as follows according to ValueType.

- kValueTypeInteger, kValueTypeUInteger, kValueTypeReal: Returns a digit sequence that represents the set value.
- kValueTypeString: Returns the set character string as-is.
- kValueTypeBoolean: Returns "false" or "true" depending on the set truth value.
- kValueTypeArray: Returns "array".
- kValueTypeObject: Returns "object".
- kValueTypeNull: Returns "null".

sce::Json::Parser Class

Summary

sce::Json::Parser

Parser class

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Parser {};
    }
}
```

Description

This is a class of only class methods that parse JSON documents.

Methods

Method	Description
<code>parse(char)</code>	Get JSON document from character string and parse
<code>parse(file)</code>	Get JSON document from file and parse
<code>parse(func)</code>	Get JSON document from callback function and parse

Type Definition

DataProvideFunction

Data provide callback function

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Parser {
            typedef int32_t (*DataProvideFunction) (
                char& data,
                void* userdata
            );
        };
    };
}
```

Arguments

data (out) 1-byte data provided to the parser
userdata (in) User-defined information specified upon callback function setting

Return Values

Return 0 when data was provided normally.
Return a negative value to interrupt the processing due to abnormalities, etc.

Description

This is an interface definition for a callback function that provides JSON document data upon parsing. When a function that follows this specification is implemented and specified as an argument upon a `parse()` call, it will be called back every time the parser attempts to read 1 byte from the JSON document. Return data of the JSON document to parse 1 byte at a time to *data*.

Return a negative value if you want to interrupt the parse processing. In such cases, `parse()` will return the value returned by the callback as-is.

Public Class Methods

parse(char)

Get JSON document from character string and parse

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Parser {
            static int32_t parse(
                Value& dst,
                const char *src,
                size_t size
            );
        };
    }
}
```

Arguments

dst (out) Value object to receive the parse results
src (in) Pointer to JSON document character string
size (in) Size of JSON document character string

Return Values

Returns SCE_OK (=0) for normal termination.

Returns an error code (negative value) for errors (refer to the [Return Codes](#) section for details).

Description

Parses the JSON document character string specified with *src* and *size*, formulates Value objects, and returns them to &*dst*.

parse(file)

Get JSON document from file and parse

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Parser {
            static int32_t parse(
                Value& dst,
                const char *path
            );
        };
    };
}
```

Arguments

dst (out) Value object to receive the parse results
path (in) JSON document file path

Return Values

Returns SCE_OK (=0) for normal termination.

Returns an error code (negative value) for errors (refer to the "[Return Codes](#)" section for details).

If SCE_JSON_ERROR_NOBUF returns, it means that 0 bytes were specified for the buffer size when the Json library was initialized. Set an appropriate size and then perform initialization.

Description

Reads the file specified with *path*, parses it as a JSON document, formulates Value objects, and returns them to &*dst*.

parse(func)

Get JSON document from callback function and parse

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Parser {
            static int32_t parse(
                Value& dst,
                DataProvideFunction func,
                void *userdata
            );
        };
    };
}
```

Arguments

<i>dst</i>	(out) Value object to receive the parse results
<i>func</i>	(in) Data provide callback function
<i>userdata</i>	(in) User-defined information to pass to the data provide callback function

Return Values

Returns SCE_OK (=0) for normal termination.

Returns an error code (negative value) for errors (refer to the "[Return Codes](#)" section for details).

Description

Parses the JSON document obtained from the data provide callback function specified with *func*, formulates Value objects, and returns them to &*dst*.

sce::Json::Initializer Interface Class

Summary

sce::Json::Initializer

Object initialize interface class

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Initializer {};
    }
}
```

Description

Before generating other `sce::Json` objects , this class must be generated and initialized.

Methods

Method	Description
Initializer()	Constructor
~Initializer()	Destructor
initialize()	Initialize
terminate()	Terminate

000004892117

Constructors and Destructors

Initializer

Constructor

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Initializer {
            Initializer();
        }
    }
}
```

Return Values

None

Description

This is a constructor.

SCE CONFIDENTIAL

~Initializer

Destructor

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Initializer {
            ~Initializer();
        }
    }
}
```

Return Values

None

Description

This is a destructor.

Public Instance Methods

initialize

Initialize

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Initializer {
            int initialize(
                const InitParameter *initParam
            );
        };
    };
}
```

Arguments

initParam (in) Initialize parameters

Return Values

Returns SCE_OK (=0) for normal termination.

Returns an error code (negative value) for errors (refer to the "[Return Codes](#)" section for details).

Description

This initializes an `Initializer` object. It must be called before using an `Initializer` object. For *initParam*, specify each public instance field such as memory allocator appropriately.

SCE CONFIDENTIAL

terminate

Terminate

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class Initializer {
            int terminate();
        }
    }
}
```

Return Values

Returns SCE_OK (=0).

Description

Terminates this `Initializer` instance and releases the occupied memory.

sce::Json::InitParameter Class

Summary

sce::Json::InitParameter

Initialize parameter class

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class InitParameter {};
    }
}
```

Description

This is the parameters to pass upon Initializer object initialization. Pass them after first setting appropriate values for each of the following public instance fields.

Fields

Public Instance Fields

MemAllocator *allocator (in) Memory allocator
void *userData (in) User-defined data to pass to the memory allocator
size_t filebuffersize (in) Buffer size upon file read

Methods

Method	Description
InitParameter()	Constructor
InitParameter(param)	Constructor (parameter specification)

Constructors and Destructors

InitParameter

Constructor

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class InitParameter {
            InitParameter();
        }
    }
}
```

Return Values

None

Description

This is a default constructor that initializes each member with 0.

InitParameter(param)

Constructor (parameter specification)

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class InitParameter {
            InitParameter(
                MemAllocator *al,
                void *ud,
                size_t fbsiz
            );
        };
    };
}
```

Arguments

<i>al</i>	(in) Pointer to the memory allocator
<i>ud</i>	(in) Pointer to the user-defined data to pass to the memory allocator
<i>fbsiz</i>	(in) Buffer size upon file read

Return Values

None

Description

This is a constructor that initializes each member using the values specified with the parameters.

sce::Json::MemAllocator Interface Class

Summary

sce::Json::MemAllocator

Memory allocator interface class

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class MemAllocator {};
    }
}
```

Description

This is a memory allocator interface class.
Inherit this class and implement a memory allocator. It will be called when memory in the library must be allocated/released.

Methods

Method	Description
MemAllocator()	Constructor
~MemAllocator()	Destructor
allocate()	Allocate memory
deallocate()	Release memory
notifyError()	Notify error

Constructors and Destructors

MemAllocator

Constructor

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class MemAllocator {
            MemAllocator();
        }
    }
}
```

Return Values

None

Description

This is a constructor.

SCE CONFIDENTIAL

~MemAllocator

Destructor

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class MemAllocator {
            virtual ~MemAllocator();
        }
    }
}
```

Return Values

None

Description

This is a destructor.

Public Instance Methods

allocate

Allocate memory

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class MemAllocator {
            virtual void *allocate(
                size_t size,
                void *userData
            )=0;
        };
    };
}
```

Arguments

size (in) Size
userData (in) User-defined data set to `InitParameter::userData`

Return Values

Return the start pointer (Non-NULL) to the allocated memory area.
 Return NULL if the memory allocation failed.

Description

This is a memory allocating function.

Allocate memory and return the pointer.

When NULL is returned upon failing to allocate memory, the `SCE_JSON_ERROR_NOMEM` error will be notified in `notifyError()`.

For details, refer to `notifyError()`.

SCE CONFIDENTIAL

deallocate

Release memory

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class MemAllocator {
            virtual void deallocate(
                void *ptr,
                void *userData
            )=0;
        };
    };
}
```

Arguments

ptr (in) Pointer to release
userData (in) User-defined data set to `InitParameter::userData`

Return Values

None

Description

This function releases memory.

SCE CONFIDENTIAL

notifyError

Notify an error

Definition

```
#include <json.h>
namespace sce {
    namespace Json {
        class MemAllocator {
            virtual void notifyError(
                int32_t error,
                size_t size;
                void* userData
            );
        };
    }
}
```

Arguments

error (in) Error code
size (in) Size specified upon allocate
userData (in) User defined data set to InitParameter::userData

Return Values

None

Description

This method notifies the error that occurred with MemAllocator.

Currently, the only error that is notified is SCE_JSON_ERROR_NOMEM.

Because an error will be received when memory allocation cannot be carried out in operating a String, Array, Object, or Value object, override and implement appropriate processing.

Implementation in the base class is as follows.

```
switch(error) {
default:
    fprintf(stderr, "[libSceJson::MemAllocator]unkown
nofication[%#x]\n", error);
    break;
SCE_JSON_ERROR_NOMEM:
    fprintf( stderr, "[libSceJson::MemAllocator]Allocation Fail size
= %ld\n", size);
    break;
}
```