# Contents

## Namespace Documentation

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Bse Namespace Reference</td>
<td>6</td>
</tr>
<tr>
<td>1.2</td>
<td>Bse::AnsiColors Namespace Reference</td>
<td>47</td>
</tr>
<tr>
<td>1.3</td>
<td>Bse::Lib Namespace Reference</td>
<td>49</td>
</tr>
<tr>
<td>1.4</td>
<td>Bse::Path Namespace Reference</td>
<td>49</td>
</tr>
<tr>
<td>1.5</td>
<td>Bse::Procedure Namespace Reference</td>
<td>54</td>
</tr>
<tr>
<td>1.6</td>
<td>Bse::Re Namespace Reference</td>
<td>54</td>
</tr>
<tr>
<td>1.7</td>
<td>Bse::Test Namespace Reference</td>
<td>54</td>
</tr>
<tr>
<td>1.8</td>
<td>Bse::Xms Namespace Reference</td>
<td>56</td>
</tr>
<tr>
<td>1.9</td>
<td>Sfi Module Reference</td>
<td>56</td>
</tr>
</tbody>
</table>

## Class Documentation

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Bse::AlignedArray&lt; T, ALIGNMENT &gt; Class Template Reference</td>
<td>58</td>
</tr>
<tr>
<td>2.2</td>
<td>Bse::AlignedPOD&lt; SIZE &gt; Struct Template Reference</td>
<td>58</td>
</tr>
<tr>
<td>2.3</td>
<td>Bse::AsyncBlockingQueue&lt; Value &gt; Class Template Reference</td>
<td>58</td>
</tr>
<tr>
<td>2.4</td>
<td>Bse::AutoSeeder Class Reference</td>
<td>59</td>
</tr>
<tr>
<td>2.5</td>
<td>Bse::AuxData Struct Reference</td>
<td>60</td>
</tr>
<tr>
<td>2.6</td>
<td>Bse::AuxDataSeq Struct Reference</td>
<td>60</td>
</tr>
<tr>
<td>2.7</td>
<td>Bse::Blob Class Reference</td>
<td>60</td>
</tr>
<tr>
<td>2.8</td>
<td>Bse::Bus Interface Reference</td>
<td>62</td>
</tr>
<tr>
<td>2.9</td>
<td>Bse::BusInterface Class Reference</td>
<td>64</td>
</tr>
<tr>
<td>2.10</td>
<td>Bse::BusImpl Class Reference</td>
<td>66</td>
</tr>
<tr>
<td>2.11</td>
<td>Bse::Category Struct Reference</td>
<td>66</td>
</tr>
<tr>
<td>2.12</td>
<td>Bse::CategorySeq Struct Reference</td>
<td>67</td>
</tr>
<tr>
<td>2.13</td>
<td>Bse::Container Interface Reference</td>
<td>67</td>
</tr>
<tr>
<td>2.14</td>
<td>Bse::ContainerIface Class Reference</td>
<td>68</td>
</tr>
<tr>
<td>2.15</td>
<td>Bse::ContainerImpl Class Reference</td>
<td>70</td>
</tr>
<tr>
<td>2.16</td>
<td>Bse::ContextMerger Interface Reference</td>
<td>70</td>
</tr>
<tr>
<td>2.17</td>
<td>Bse::ContextMergerIface Class Reference</td>
<td>71</td>
</tr>
<tr>
<td>2.18</td>
<td>Bse::ContextMergerImpl Class Reference</td>
<td>73</td>
</tr>
<tr>
<td>2.19</td>
<td>ConvertAny Struct Reference</td>
<td>74</td>
</tr>
<tr>
<td>2.20</td>
<td>Bse::CSynth Interface Reference</td>
<td>74</td>
</tr>
<tr>
<td>2.21</td>
<td>Bse::CSynthIface Class Reference</td>
<td>75</td>
</tr>
<tr>
<td>2.22</td>
<td>Bse::CSynthImpl Class Reference</td>
<td>76</td>
</tr>
<tr>
<td>2.23</td>
<td>Bse::Xms::DataConverter&lt; T, typename &gt; Struct Template Reference</td>
<td>76</td>
</tr>
<tr>
<td>2.24</td>
<td>Bse::DataKey&lt; Type &gt; Class Template Reference</td>
<td>77</td>
</tr>
<tr>
<td>2.25</td>
<td>Bse::DataList Class Reference</td>
<td>77</td>
</tr>
<tr>
<td>2.26</td>
<td>Bse::DataListContainer Class Reference</td>
<td>78</td>
</tr>
<tr>
<td>2.27</td>
<td>Bse::Device Interface Reference</td>
<td>80</td>
</tr>
<tr>
<td>2.28</td>
<td>Bse::DeviceCrawlerIface Class Reference</td>
<td>82</td>
</tr>
<tr>
<td>2.29</td>
<td>Bse::DeviceCrawlerImpl Class Reference</td>
<td>83</td>
</tr>
<tr>
<td>2.30</td>
<td>Bse::DeviceIface Class Reference</td>
<td>83</td>
</tr>
<tr>
<td>2.31</td>
<td>Bse::DeviceImpl Class Reference</td>
<td>85</td>
</tr>
<tr>
<td>2.32</td>
<td>Bse::DeviceTypeInfo Struct Reference</td>
<td>86</td>
</tr>
<tr>
<td>2.33</td>
<td>Bse::DriverEntry Struct Reference</td>
<td>86</td>
</tr>
<tr>
<td>2.34</td>
<td>Bse::DriverEntrySeq Struct Reference</td>
<td>86</td>
</tr>
<tr>
<td>2.35</td>
<td>Bse::EditableSample Interface Reference</td>
<td>86</td>
</tr>
<tr>
<td>2.36</td>
<td>Bse::EditableSampleIface Class Reference</td>
<td>89</td>
</tr>
<tr>
<td>Section</td>
<td>Class/Reference</td>
<td>Page</td>
</tr>
<tr>
<td>----------</td>
<td>---------------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>2.37</td>
<td>Bse::EditableSampleImpl Class Reference</td>
<td>90</td>
</tr>
<tr>
<td>2.38</td>
<td>Bse::Flac1Handle Class Reference</td>
<td>91</td>
</tr>
<tr>
<td>2.39</td>
<td>Bse::FloatSeq Struct Reference</td>
<td>92</td>
</tr>
<tr>
<td>2.40</td>
<td>Bse::FriendAllocator&lt; T &gt; Class Template Reference</td>
<td>92</td>
</tr>
<tr>
<td>2.41</td>
<td>Bse::Icon Struct Reference</td>
<td>93</td>
</tr>
<tr>
<td>2.42</td>
<td>Bse::Item Interface Reference</td>
<td>94</td>
</tr>
<tr>
<td>2.43</td>
<td>Bse::ItemImpl Class Reference</td>
<td>98</td>
</tr>
<tr>
<td>2.44</td>
<td>Bse::ItemImpl Class Reference</td>
<td>100</td>
</tr>
<tr>
<td>2.45</td>
<td>Bse::ItemSeq Struct Reference</td>
<td>102</td>
</tr>
<tr>
<td>2.46</td>
<td>Bse::KeccakCryptoRng Class Reference</td>
<td>102</td>
</tr>
<tr>
<td>2.47</td>
<td>Bse::Lib::KeccakF1600 Class Reference</td>
<td>104</td>
</tr>
<tr>
<td>2.48</td>
<td>Bse::KeccakFastRng Class Reference</td>
<td>105</td>
</tr>
<tr>
<td>2.49</td>
<td>Bse::KeccakGoodRng Class Reference</td>
<td>106</td>
</tr>
<tr>
<td>2.50</td>
<td>Bse::KeccakRng Class Reference</td>
<td>108</td>
</tr>
<tr>
<td>2.51</td>
<td>Bse::LegacyObject Interface Reference</td>
<td>112</td>
</tr>
<tr>
<td>2.52</td>
<td>Bse::LegacyObjectIface Class Reference</td>
<td>114</td>
</tr>
<tr>
<td>2.53</td>
<td>Bse::LegacyObjectImpl Class Reference</td>
<td>116</td>
</tr>
<tr>
<td>2.54</td>
<td>Bse::MidiNotifier Interface Reference</td>
<td>116</td>
</tr>
<tr>
<td>2.55</td>
<td>Bse::MidiNotifierIface Class Reference</td>
<td>118</td>
</tr>
<tr>
<td>2.56</td>
<td>Bse::MidiNotifierIml Class Reference</td>
<td>119</td>
</tr>
<tr>
<td>2.57</td>
<td>Bse::MidiSynth Interface Reference</td>
<td>119</td>
</tr>
<tr>
<td>2.58</td>
<td>Bse::MidiSynthIface Class Reference</td>
<td>121</td>
</tr>
<tr>
<td>2.59</td>
<td>Bse::MidiSynthIml Class Reference</td>
<td>122</td>
</tr>
<tr>
<td>2.60</td>
<td>Bse::Module Class Reference</td>
<td>122</td>
</tr>
<tr>
<td>2.61</td>
<td>Bse::ModuleIface Class Reference</td>
<td>124</td>
</tr>
<tr>
<td>2.62</td>
<td>Bse::ModuleImpl Class Reference</td>
<td>126</td>
</tr>
<tr>
<td>2.63</td>
<td>Bse::ModuleTypeInfo Struct Reference</td>
<td>127</td>
</tr>
<tr>
<td>2.64</td>
<td>Bse::NoteDescription Struct Reference</td>
<td>127</td>
</tr>
<tr>
<td>2.65</td>
<td>Bse::Object Interface Reference</td>
<td>127</td>
</tr>
<tr>
<td>2.66</td>
<td>Bse::ObjectIface Class Reference</td>
<td>129</td>
</tr>
<tr>
<td>2.67</td>
<td>Bse::ObjectIml Class Reference</td>
<td>130</td>
</tr>
<tr>
<td>2.68</td>
<td>Bse::Part Interface Reference</td>
<td>130</td>
</tr>
<tr>
<td>2.69</td>
<td>Bse::PartControl Struct Reference</td>
<td>137</td>
</tr>
<tr>
<td>2.70</td>
<td>Bse::PartControlSeq Struct Reference</td>
<td>137</td>
</tr>
<tr>
<td>2.71</td>
<td>Bse::Partface Class Reference</td>
<td>138</td>
</tr>
<tr>
<td>2.72</td>
<td>Bse::PartImpl Class Reference</td>
<td>139</td>
</tr>
<tr>
<td>2.73</td>
<td>Bse::PartLink Struct Reference</td>
<td>140</td>
</tr>
<tr>
<td>2.74</td>
<td>Bse::PartLinkSeq Struct Reference</td>
<td>140</td>
</tr>
<tr>
<td>2.75</td>
<td>Bse::PartNote Struct Reference</td>
<td>140</td>
</tr>
<tr>
<td>2.76</td>
<td>Bse::PartNoteSeq Struct Reference</td>
<td>140</td>
</tr>
<tr>
<td>2.77</td>
<td>Bse::PartSeq Struct Reference</td>
<td>140</td>
</tr>
<tr>
<td>2.78</td>
<td>Bse::Pcg32Rng Class Reference</td>
<td>141</td>
</tr>
<tr>
<td>2.79</td>
<td>Bse::PcmWriter Interface Reference</td>
<td>142</td>
</tr>
<tr>
<td>2.80</td>
<td>Bse::PcmWriterIface Class Reference</td>
<td>144</td>
</tr>
<tr>
<td>2.81</td>
<td>Bse::PcmWriterImpl Class Reference</td>
<td>145</td>
</tr>
<tr>
<td>2.82</td>
<td>Bse::PixelSeq Struct Reference</td>
<td>145</td>
</tr>
<tr>
<td>2.83</td>
<td>Bse::ProbeFeatures Struct Reference</td>
<td>146</td>
</tr>
<tr>
<td>2.84</td>
<td>Bse::Project Interface Reference</td>
<td>146</td>
</tr>
<tr>
<td>2.85</td>
<td>Bse::ProjectIface Class Reference</td>
<td>153</td>
</tr>
<tr>
<td>2.86</td>
<td>Bse::ProjectIml Class Reference</td>
<td>155</td>
</tr>
<tr>
<td>2.87</td>
<td>Bse::PropertyCandidates Struct Reference</td>
<td>156</td>
</tr>
<tr>
<td>2.88</td>
<td>Bse::Xms::SerializationNode::QueuedArgs Struct Reference</td>
<td>156</td>
</tr>
<tr>
<td>2.89</td>
<td>Bse::Xms::Reflink Class Reference</td>
<td>156</td>
</tr>
<tr>
<td>2.90</td>
<td>Bse::Resampler2 Class Reference</td>
<td>156</td>
</tr>
<tr>
<td>2.91</td>
<td>Bse::SampleFileInfo Struct Reference</td>
<td>156</td>
</tr>
<tr>
<td>2.92</td>
<td>Bse::Lib::ScopedLocale Class Reference</td>
<td>158</td>
</tr>
<tr>
<td>2.93</td>
<td>Bse::Lib::ScopedPosixLocale Class Reference</td>
<td>159</td>
</tr>
<tr>
<td>2.94</td>
<td>Bse::Sequencer Class Reference</td>
<td>160</td>
</tr>
<tr>
<td>2.95</td>
<td>Bse::Xms::SerializableInterface Class Reference</td>
<td>161</td>
</tr>
<tr>
<td>Section</td>
<td>Reference</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>-----------</td>
<td>------</td>
</tr>
<tr>
<td>2.100</td>
<td>Bse::ServerImpl Class Reference</td>
<td>177</td>
</tr>
<tr>
<td>2.101</td>
<td>SfRecFields Struct Reference</td>
<td>178</td>
</tr>
<tr>
<td>2.102</td>
<td>Bse::SHA3_224 Struct Reference</td>
<td>178</td>
</tr>
<tr>
<td>2.103</td>
<td>Bse::SHA3_256 Struct Reference</td>
<td>179</td>
</tr>
<tr>
<td>2.104</td>
<td>Bse::SHA3_384 Struct Reference</td>
<td>180</td>
</tr>
<tr>
<td>2.105</td>
<td>Bse::SHA3_512 Struct Reference</td>
<td>181</td>
</tr>
<tr>
<td>2.106</td>
<td>Bse::SHAKE128 Struct Reference</td>
<td>182</td>
</tr>
<tr>
<td>2.107</td>
<td>Bse::SHAKE256 Struct Reference</td>
<td>183</td>
</tr>
<tr>
<td>2.108</td>
<td>Bse::SharedMemory Struct Reference</td>
<td>184</td>
</tr>
<tr>
<td>2.109</td>
<td>Bse::ShmFragment Struct Reference</td>
<td>185</td>
</tr>
<tr>
<td>2.110</td>
<td>Bse::ShmFragmentSeq Struct Reference</td>
<td>185</td>
</tr>
<tr>
<td>2.111</td>
<td>Bse::SignalMonitor Interface Reference</td>
<td>185</td>
</tr>
<tr>
<td>2.112</td>
<td>Bse::SignalMonitorIface Class Reference</td>
<td>188</td>
</tr>
<tr>
<td>2.113</td>
<td>Bse::SignalMonitorImpl Class Reference</td>
<td>189</td>
</tr>
<tr>
<td>2.114</td>
<td>Bse::SNet Interface Reference</td>
<td>189</td>
</tr>
<tr>
<td>2.115</td>
<td>Bse::SNetIFace Class Reference</td>
<td>191</td>
</tr>
<tr>
<td>2.116</td>
<td>Bse::SNetImpl Class Reference</td>
<td>193</td>
</tr>
<tr>
<td>2.117</td>
<td>Bse::Song Interface Reference</td>
<td>194</td>
</tr>
<tr>
<td>2.118</td>
<td>Bse::SongIFace Class Reference</td>
<td>197</td>
</tr>
<tr>
<td>2.119</td>
<td>Bse::SongImpl Class Reference</td>
<td>199</td>
</tr>
<tr>
<td>2.120</td>
<td>Bse::SongTiming Struct Reference</td>
<td>200</td>
</tr>
<tr>
<td>2.121</td>
<td>Bse::SoundFont Interface Reference</td>
<td>200</td>
</tr>
<tr>
<td>2.122</td>
<td>Bse::SoundFontIface Class Reference</td>
<td>201</td>
</tr>
<tr>
<td>2.123</td>
<td>Bse::SoundFontImpl Class Reference</td>
<td>202</td>
</tr>
<tr>
<td>2.124</td>
<td>Bse::SoundFontRepoIFace Class Reference</td>
<td>202</td>
</tr>
<tr>
<td>2.125</td>
<td>Bse::SoundFontRepoImpl Class Reference</td>
<td>204</td>
</tr>
<tr>
<td>2.126</td>
<td>Bse::Source Interface Reference</td>
<td>204</td>
</tr>
<tr>
<td>2.127</td>
<td>Bse::SourceIFace Class Reference</td>
<td>210</td>
</tr>
<tr>
<td>2.128</td>
<td>Bse::SourceImpl Class Reference</td>
<td>212</td>
</tr>
<tr>
<td>2.129</td>
<td>Bse::Spinlock Class Reference</td>
<td>212</td>
</tr>
<tr>
<td>2.130</td>
<td>Bse::Lib::StringFormatter Class Reference</td>
<td>213</td>
</tr>
<tr>
<td>2.131</td>
<td>Bse::Strings Class Reference</td>
<td>214</td>
</tr>
<tr>
<td>2.132</td>
<td>Bse::StringSeq Struct Reference</td>
<td>215</td>
</tr>
<tr>
<td>2.133</td>
<td>Bse::SubSynth Interface Reference</td>
<td>215</td>
</tr>
<tr>
<td>2.134</td>
<td>Bse::SubSynthIface Class Reference</td>
<td>217</td>
</tr>
<tr>
<td>2.135</td>
<td>Bse::SubSynthImpl Class Reference</td>
<td>218</td>
</tr>
<tr>
<td>2.136</td>
<td>Bse::Super Interface Reference</td>
<td>219</td>
</tr>
<tr>
<td>2.137</td>
<td>Bse::SuperIFace Class Reference</td>
<td>219</td>
</tr>
<tr>
<td>2.138</td>
<td>Bse::SuperImpl Class Reference</td>
<td>221</td>
</tr>
<tr>
<td>2.139</td>
<td>Bse::SuperSeq Struct Reference</td>
<td>221</td>
</tr>
<tr>
<td>2.140</td>
<td>Bse::TaskRegistry Class Reference</td>
<td>222</td>
</tr>
<tr>
<td>2.141</td>
<td>Bse::TaskStatus Struct Reference</td>
<td>223</td>
</tr>
<tr>
<td>2.142</td>
<td>Bse::Test::Timer Class Reference</td>
<td>225</td>
</tr>
<tr>
<td>2.143</td>
<td>Bse::Track Interface Reference</td>
<td>226</td>
</tr>
<tr>
<td>2.144</td>
<td>Bse::TrackIFace Class Reference</td>
<td>230</td>
</tr>
<tr>
<td>2.145</td>
<td>Bse::TrackImpl Class Reference</td>
<td>232</td>
</tr>
<tr>
<td>2.146</td>
<td>Bse::TrackPart Struct Reference</td>
<td>233</td>
</tr>
<tr>
<td>2.147</td>
<td>Bse::TrackPartSeq Struct Reference</td>
<td>233</td>
</tr>
<tr>
<td>2.148</td>
<td>Bse::TrackSeq Struct Reference</td>
<td>233</td>
</tr>
<tr>
<td>2.149</td>
<td>Bse::ItemImpl::UndoDescriptor&lt; Obj &gt; Class Template Reference</td>
<td>233</td>
</tr>
<tr>
<td>2.150</td>
<td>Bse::UserMessage Struct Reference</td>
<td>233</td>
</tr>
<tr>
<td>2.151</td>
<td>Bse::Wave Interface Reference</td>
<td>235</td>
</tr>
<tr>
<td>2.152</td>
<td>Bse::WaveIFace Class Reference</td>
<td>236</td>
</tr>
<tr>
<td>2.153</td>
<td>Bse::WaveImpl Class Reference</td>
<td>238</td>
</tr>
<tr>
<td>2.154</td>
<td>Bse::WaveOsc Interface Reference</td>
<td>239</td>
</tr>
</tbody>
</table>
CONTENTS

2.155 Bse::WaveOsciFace Class Reference ................................................. 240
2.156 Bse::WaveOsciImpl Class Reference .................................................. 242
2.157 Bse::WaveOscSeq Struct Reference ..................................................... 242
2.158 Bse::WaveRepo Interface Reference .................................................... 243
2.159 Bse::WaveRepoIface Class Reference .................................................. 244
2.160 Bse::WaveRepoImpl Class Reference ................................................... 246

3 File Documentation

3.1 bse/bseapi.idl File Reference .............................................................. 247

Bibliography

Index
Chapter 1

Namespace Documentation

1.1 Bse Namespace Reference

The Bse namespace contains all functions of the synthesis engine.

Namespaces

- **AnsiColors**
  - The AnsiColors namespace contains utility functions for colored terminal output.
- **Lib**
  - Namespace for implementation internals.
- **Path**
  - The Path namespace provides functions for file path manipulation and testing.
- **Procedure**
  - The Procedure namespace contains procedure/IDL helpers.
- **Re**
  - Some std::regex wrappers to simplify usage and reduce compilation time.
- **Test**
  - The Test namespace offers utilities for unit tests.
- **Xms**
  - Namespace for XML based serialization.

Classes

- **class AlignedArray**
  - Class to maintain an array of aligned memory.
- **struct AlignedPOD**
  - Helper to provide memory for placement new AlignedPOD<SIZE> is aligned like max_align_t or like malloc()-ed memory and provides SIZE bytes.
- **class AsyncBlockingQueue**
  - Asynchronous queue to push/pop values across thread boundaries.
- **class AutoSeeder**
  - AutoSeeder provides non-deterministic seeding entropy.
- **struct AuxData**
  - AuxData - record to describe entity attributes with "key = value" strings.
- **struct AuxDataSeq**
  - AuxDataSeq - a variable length list of AuxData records.
- **class Blob**
  - Binary large object storage container.
- **interface Bus**
  - Interface for effect stacks and per-track audio signal routing to the master output.
• class BusIface
  IDL interface class for Bse::Bus.
• class BusImpl
• struct Category
  Categories describe useful type entities.
• struct CategorySeq
  Sequence of Category records.
• interface Container
  Base interface type for containers of Item derived types.
• class ContainerIface
  IDL interface class for Bse::Container.
• class ContainerImpl
• interface ContextMerger
  Source module for merging multiple synthesis contexts, used to implement polyphony.
• class ContextMergerIface
  IDL interface class for Bse::ContextMerger.
• class ContextMergerImpl
• interface CSynth
  Customizable synthesis (filter) network container.
• class CSynthIface
  IDL interface class for Bse::CSynth.
• class CSynthImpl
• class DataKey
  DataKey objects are used to identify and manage custom data members of DataListContainer objects.
• class DataList
  Underlying storage implementation for a DataListContainer.
• class DataListContainer
  DataListContainer - typesafe storage and retrieval of arbitrary members.
• interface Device
  Interface for the encapsulation of audio processors.
• class DeviceCrawlerIface
  IDL interface class for Bse::DeviceCrawler.
• class DeviceCrawlerImpl
• class DeviceIface
  IDL interface class for Bse::Device.
• class DeviceImpl
• struct DeviceTypeInfo
  Info for device types.
• struct DriverEntry
  Driver information for PCM and MIDI handling.
• struct DriverEntrySeq
  DriverEntry sequence.
• interface EditableSample
  Interface for editable PCM wave samples.
• class EditableSampleIface
  IDL interface class for Bse::EditableSample.
• class EditableSampleImpl
• class Flac1Handle
  Flac1Handle supports storing flac files as binary appendix to BSE files.
• struct FloatSeq
  A list of floating point values.
• class FriendAllocator
1.1 Bse Namespace Reference

A std::make_shared<> wrapper class to access private ctor & dtor.

- struct Icon
  Representation of an icon pixel image.
- interface Item
  Base interface type for objects that can be added to a container.
- class ItemIface
  IDL interface class for Bse::Item.
- class ItemImpl
- struct ItemSeq
  A list of Item or derived objects.
- class KeccakCryptoRng
  KeccakCryptoRng - A KeccakF1600 based cryptographic quality pseudo-random number generator.
- class KeccakFastRng
  KeccakFastRng - A KeccakF1600 based fast pseudo-random number generator.
- class KeccakGoodRng
  KeccakGoodRng - A KeccakF1600 based good quality pseudo-random number generator.
- class KeccakRng
  KeccakRng - A KeccakF1600 based pseudo-random number generator.
- interface LegacyObject
  Base type for all legacy objects, derived from struct BseObject.
- class LegacyObjectIface
  IDL interface class for Bse::LegacyObject.
- class LegacyObjectImpl
- interface MidiNotifier
  Interface for MIDI event notification.
- class MidiNotifierIface
  IDL interface class for Bse::MidiNotifier.
- class MidiNotifierImpl
- interface MidiSynth
  Interface for MIDI synthesis networks.
- class MidiSynthIface
  IDL interface class for Bse::MidiSynth.
- class MidiSynthImpl
- class Module
  Interface for the encapsulation of audio processors.
- class ModuleIface
  IDL interface class for Bse::Module.
- class ModuleImpl
- struct ModuleTypeInfo
  Info for module types.
- struct NoteDescription
  A note description provides all needed details about a specific note. ".
- interface Object
  Base type for all new style C++ objects.
- class ObjectIface
  IDL interface class for Bse::Object.
- class ObjectImpl
- interface Part
  Data interface for containment of piano notes and MIDI effects.
- struct PartControl
  Part specific control event representation.
- struct PartControlSeq
A list of part control events.

- class `PartInterface`
  
  IDL interface class for `Bse::Part`.

- class `PartImpl`

- struct `PartLink`
  
  Record representing the use of a Part within a Track at a specific position.

- struct `PartLinkSeq`
  
  Sequence of `PartLink` records.

- struct `PartNote`
  
  Part specific note event representation.

- struct `PartNoteSeq`
  
  A list of part note events.

- struct `PartSeq`
  
  A list of Part or derived types.

- class `Pcg32Rng`
  
  `Pcg32Rng` is a permutating linear congruential PRNG.

- interface `PcmWriter`
  
  Interface for writing PCM wave data.

- class `PcmWriterIface`
  
  IDL interface class for `Bse::PcmWriter`.

- class `PcmWriterImpl`

- struct `PixelSeq`
  
  Representation of an image pixel sequence in ARGB format.

- struct `ProbeFeatures`
  
  Bits representing a selection of probe sample data features.

- interface `Project`
  
  Projects support loading, saving, playback and act as containers for all other sound objects.

- class `ProjectIface`
  
  IDL interface class for `Bse::Project`.

- class `ProjectImpl`

- struct `PropertyCandidates`
  
  A list of items suitable to set as a specific property value.

- class `Resampler2`
  
  Interface for factor 2 resampling classes.

- struct `SampleFileInfo`
  
  Structure containing meta data for multi wave samples.

- class `Sequencer`
  
  Note and MIDI sequencer.

- interface `Server`
  
  Main Bse remote origin object.

- class `ServerIface`
  
  IDL interface class for `Bse::Server`.

- class `ServerImpl`

- struct `SHA3_224`
  
  `SHA3_224` - 224 Bit digest generation.

- struct `SHA3_256`
  
  `SHA3_256` - 256 Bit digest generation.

- struct `SHA3_384`
  
  `SHA3_384` - 384 Bit digest generation.

- struct `SHA3_512`
  
  `SHA3_512` - 512 Bit digest generation.

- struct `SHAKE128`
1.1 Bse Namespace Reference

**SHAKE128** - 128 Bit extendable output digest generation.
- struct **SHAKE256**
  - 256 Bit extendable output digest generation.

**SharedMemory**
- Descriptor for a shared memory region.

**ShmFragment**
- Fragment description for interesting bits of shared memory.

**ShmFragmentSeq**
- Collection of shared memory fragments.

**SignalMonitor**
- Interface for monitoring output signals.

**SignalMonitorIface**
- IDL interface class for Bse::SignalMonitor.

**SignalMonitorImpl**
- Base interface type for all kinds of synthesis networks.

**SNet**
- IDL interface class for Bse::SNet.

**SNetIface**
- Interface for Track and Part objects, as well as meta data for sequencing.

**SNetImpl**
- IDL interface class for Bse::SNet.

**Song**
- Interface for Track and Part objects, as well as meta data for sequencing.

**SongIface**
- IDL interface class for Bse::Song.

**SongImpl**
- Song timing configuration.

**SongTiming**
- Interface for sound fonts.

**SoundFont**
- IDL interface class for Bse::SoundFont.

**SoundFontIface**
- Base interface type for synthesis modules with input or output streams.

**SoundFontImpl**
- IDL interface class for Bse::SoundFont.

**SoundFontRepo**
- Interface Source
- Base interface type for Item managers.

**Spinlock**
- The Spinlock uses low-latency busy spinning to acquire locks.

**Strings**
- Convenience Constructor for StringSeq or std::vector< std::string >

**StringSeq**
- StringSeq - a variable length list of test strings.

**SubSynth**
- Synthesizer module for embedding (rerouting input and output) of another synthesizer network (SNet).

**SubSynthIface**
- IDL interface class for Bse::SubSynth.

**SubSynthImpl**
- Base interface type for Item managers.
• class SuperIface
  IDL interface class for Bse::Super.
• class SuperImpl
• struct SuperSeq
  A list of Super type objects.
• class TaskRegistry
  The task registry keeps track of runtime threads for profiling and statistical purposes.
• struct TaskStatus
  Acquire information about a task (process or thread) at runtime.
• interface Track
  Interface for sequencing information and links to Part objects.
• class TrackIface
  IDL interface class for Bse::Track.
• class TrackImpl
• struct TrackPart
  Structure linking to a Track from within a Part.
• struct TrackPartSeq
  Sequence of TrackPart records.
• struct TrackSeq
  Sequence of Track objects.
• struct UserMessage
  Structure for submission of user interface messages from BSE.
• interface Wave
  Interface for PCM wave samples.
• class WaveIface
  IDL interface class for Bse::Wave.
• class WaveImpl
• interface WaveOsc
  Oscillator module for wave files.
• class WaveOscIface
  IDL interface class for Bse::WaveOsc.
• class WaveOscImpl
• struct WaveOscSeq
  A list of part note events.
• interface WaveRepo
  Interface serving as container for Wave objects.
• class WaveRepoIface
  IDL interface class for Bse::WaveRepo.
• class WaveRepoImpl

Typedefs
• typedef uint8_t uint8
  An 8-bit unsigned integer.
• typedef uint16_t uint16
  A 16-bit unsigned integer.
• typedef uint32_t uint32
  A 32-bit unsigned integer.
• typedef uint64_t uint64
  A 64-bit unsigned integer, use PRI+64 in format strings.
• typedef int8_t int8
  An 8-bit signed integer.
• typedef int16_t int16
  A 16-bit signed integer.
• typedef int32_t int32
  A 32-bit signed integer.
• typedef int64_t int64
  A 64-bit unsigned integer, use PRI+64 in format strings.
• typedef uint32_t uchar
  A 32-bit unsigned integer used for Unicode characters.
• typedef std::string String
  Convenience alias for std::string.
• using StringVector = vector<String>
  Convenience alias for a std::vector<std::string>.
• typedef uint32_t uint
  Provide 'uint' as convenience type.

Enumerations

• enum MonitorField { F64_GENERATION, F32_MIN, F32_MAX, F32_DB_SPL, F32_DB_TIP, END_BYTE }
  Offsets for signal monitoring fields in bytes, field type and size is used as prefix.
• enum UserMessageType { ERROR, WARNING, INFO, DEBUG }
• enum SongTelemetry { I32_TICK_POINTER, BYTECOUNT }
  Offsets for signal monitoring fields in bytes, field type and size is used as prefix.
• enum ProjectState { INACTIVE, ACTIVE, PLAYING }
  Enumeration describing the current activation and playback state of a project.
• enum ModuleFlag { ModuleFlag::NORMAL, ModuleFlag::CHEAP, ModuleFlag::EXPENSIVE, ModuleFlag::VIRTUAL_ }

Functions

• bool print_backtrace (const char *file, int line, const char *func)
  Print a C++ backtrace to stdout/stderr.
• int fmsb (uint64 val)
  The fmsb() function returns the position of the most significant bit set in the word val.
• void * aligned_alloc ( size_t total_size, size_t alignment, uint8 **free_pointer)
  Allocate a block of memory aligned to at least alignment bytes.
• void aligned_free (uint8 **free_pointer)
  Release a block of memory allocated through aligned_malloc().
• String feature_toggle_find (const String &config, const String &feature, const String &fallback)
  Find feature in config, return its value or fallback.
• bool feature_toggle_bool (const char *config, const char *feature)
  Check for feature in config, if feature is empty, checks for any feature.
• bool feature_check (const char *feature)
  Check if feature is enabled via $BSE_FEATURE.
• bool url_show (const char *url)
  Find a suitable WWW user agent (taking user configurations into account) and start it to display url.
• bool debug_key_enabled (const char *conditional)
  Check if conditional is enabled by $BSE_DEBUG.
• bool debug_key_enabled (const ::std::string &conditional)
  Check if conditional is enabled by $BSE_DEBUG.
• ::std::string debug_key_value (const char *conditional)
  Retrieve the value assigned to debug key conditional in $BSE_DEBUG.
1.1 Bse Namespace Reference

- **void diag_abort_hook** (const std::function<void(const std::string &)> &hook)
  
  Call hook for fatal_error() and diag_failed_assert().

- **template<class... Args> String string_format** (const char *format, const Args &...args)
  
  Formatted printing ala printf() into a String, using the POSIX/C locale.

- **template<class ... Args> String String void fatal_error** (const char *format, const Args &...args)
  
  Issue a printf-like message and abort the program, this function will not return.

- **template<class... Args> void warning** (const char *format, const Args &...args)
  
  Issue a printf-like warning message.

- **template<class... Args> void info** (const char *format, const Args &...args)
  
  Issue an informative printf-like message.

- **template<class... Args> void printout** (const char *format, const Args &...args)
  
  Print a message on stdout (and flush stdout) ala printf(), using the POSIX/C locale.

- **bool debug_enabled** ()
  
  Check if any kind of debugging is enabled by $BSE_DEBUG.

- **template<class... Args> void debug** (const char *cond, const char *format, const Args &...args)
  
  Issue a printf-like debugging message if cond is enabled by $BSE_DEBUG.

- **template<class... Args> void debug_message** (const char *file, int line, const char *func, const char *cond, const char *format, const Args &...args)
  
  Issue a printf-like debugging message if cond is enabled by $BSE_DEBUG.

- **template<typename RandIter, class Cmp, typename Arg> std::pair<RandIter, bool> binary_lookup_insertion_pos** (RandIter begin, RandIter end, Cmp cmp, Arg &arg)
  
  Perform a binary lookup to find the insertion position for a new element.

- **template<typename RandIter, class Cmp, typename Arg> RandIter binary_lookup_sibling** (RandIter begin, RandIter end, Cmp cmp_elements, const Arg &arg)
  
  Perform a binary lookup to yield exact or nearest match.

- **template<class ... Args> void fatal_error** (const char *format, const Args &...args)
  
  Issue a printf-like message and abort the program, this function will not return.

- **template<class... Args> void printerr** (const char *format, const Args &...args)
  
  Print a message on stderr (and flush stderr) ala printf(), using the POSIX/C locale.

- **void zintern_free** (uint8 *dc_data)
  
  Free data returned from zintern_decompress().
1.1 Bse Namespace Reference

- **uint8 * zintern_decompress** (unsigned int decompressed_size, const unsigned char *cdata, unsigned int cdata_size)
  
  **Decompress data via zlib.**

- **ServerHandle init_server_instance ()**
  
  **Retrieve a handle for the Bse::Server instance managing the Bse thread.**

- **uint exec_now (const std::function < bool() > &function)**
  
  **Run function immediately with the next event loop iteration, return true to keep alive.**

- **uint exec_now (const std::function < void() > &function)**
  
  **Run function immediately with the next event loop iteration.**

- **uint exec_timeout (const std::function < bool() > &function, uint delay_ms)**
  
  **Run function after delay_ms milliseconds have passed, return true to keep alive.**

- **uint exec_timeout (const std::function < void() > &function, uint delay_ms)**
  
  **Run function after delay_ms milliseconds have passed.**

- **bool exec_handler_clear (uint id)**
  
  **Remove a function previously added with exec_now() or exec_timeout()**

- **SfiGlueContext * init_glue_context (const gchar * client, const std::function < void() > &caller_wakeup)**
  
  **Create SFI glue layer context.**

- **void init_async (int * argc, char **argv, const char * app_name, const StringVector &args)**
  
  **Initialize and start BSE.**

- **bool init_needed ()**
  
  **Check whether init_async() still needs to be called.**

- **Icon icon_from_pixstream (const uint8 * pixstream)**
  
  **Create a Bse::Icon from a GdkPixbuf pixstream.**

- **bool icon_sanitize (Icon * icon)**
  
  **Ensure consistency of the icon fields.**

- **constexpr bool constexpr_equals (const char * a, const char * b, size_t n)**
  
  **Test string equality at compile time.**

- **template<class Type , class ... Ts> void new_inplace (Type &typemem, Ts &... args)**
  
  **Call inplace new operator by automatically inferring the Type.**

- **template<class Type > void delete_inplace (Type &typemem)**
  
  **Call inplace delete operator by automatically inferring the Type.**

- **template<class Target, class Source > std::shared_ptr < typename std::remove_pointer < Target >::type > shared_ptr_cast (Source * object)**
  
  **Shorthand for std::dynamic_pointer_cast<shared_from_this>().**

- **template<class Target, class Source > const std::shared_ptr < typename std::remove_pointer < Target >::type > shared_ptr_cast (const Source * object)**
  
  **See shared_ptr_cast(Source*).**

- **template<class Target, class Source > std::shared_ptr < Target >::type > shared_ptr_cast (std::shared_ptr < Source > &sptr)**
  
  **See shared_ptr_cast(Source*).**

- **template<class Target, class Source > const std::shared_ptr < Target >::type > shared_ptr_cast (const std::shared_ptr < Source > &sptr)**
  
  **See shared_ptr_cast(Source*).**

- **void collect_runtime_entropy (uint64 *data, size_t n)**
  
  **To provide good quality random number seeds, this function gathers entropy from a variety of process specific sources.**

- **void collect_system_entropy (uint64 *data, size_t n)**
  
  **This function adds to collect_runtime_entropy() by collecting entropy from additional but potentially slower system sources, such as interrupt counters, disk + network statistics, system load, execution + pipelining + scheduling latencies and device MACs.**
• MemoryArea find_memory_area (uint32 mem_id)
  
  Lookup a previously created memory area.

• MemoryArea create_memory_area (uint32 mem_size, uint32 alignment = BSE_CACHE_LINE_ALIGNME−
  NT)
  
  Create isolated memory area, the MemoryArea.mem_id can be used for allocate_aligned_block().

• AlignedBlock allocate_aligned_block (uint32 mem_id, uint32 length)
  
  Create a memory block from memory area mem_id, if 0, uses the internal cache-line aligned pool.

• void release_aligned_block (const AlignedBlock &block)
  
  Reallocate a previously allocated block.

• std::string runpath (RPath rpath)
  
  Retrieve various resource paths at runtime.

• const char * _(const char + string)
  
  Translate message strings in the BEAST/BSE text domain.

• std::string() _(const std::string &string)
  
  Translate message strings in the BEAST/BSE text domain.

• const char * _(const string, const char *plural, int64 t n)
  
  Translate message strings in the BEAST/BSE text domain, use forms if
  != 1.

• std::string() _(const std::string &string, const std::string &plural, int64 t n)
  
  Translate message strings in the BEAST/BSE text domain, use forms if
  != 1.

• std::string cpu_arch ()
  
  Retrieve string identifying the CPU architecture.

• String cpu_info ()

  The returned string contains: number of online CPUs, a string describing the CPU architecture, the vendor and finally a
  number of flag words describing CPU features plus a trailing space.

• uint64 timestamp_startup ()
  
  Provides the timestamp_realtime() value from program startup.

• uint64 timestamp_realtime ()
  
  Return the current time as uint64 in µseconds.

• uint64 timestamp_resolution ()
  
  Provide resolution of timestamp_benchmark() in nano-seconds.

• uint64 timestamp_benchmark ()
  
  Returns benchmark timestamp in nano-seconds, clock starts around program startup.

• String timestamp_format (uint64 stamp, uint maxlength)
  
  Convert stamp into a string, adding µsecond fractions if space permits.

• uint64 monotonic_counter ()
  
  A monotonically increasing counter, increments are atomic and visible in all threads.

• std::string executable_path ()
  
  Retrieve the path to the currently running executable.

• std::string executable_name ()
  
  Retrieve the name part of executable_path().

• std::string version ()
  
  Provide a string containing the BSE library build.

• String program_alias ()
  
  Retrieve the program name as used for logging or debug messages.

• void program_alias_init (String customname)
  
  Set program_alias to a non-localized alias other than program_argv0 if desired.

• String application_name ()
  
  Retrieve the localized program name intended for user display.

• void application_name_init (String desktopname)
  
  Set the application_name to a name other than program_alias if desired.
1.1 Bse Namespace Reference

- **String program_cwd ()**
  
  The current working directory during startup.
- **void breakpoint ()**
  
  Cause a debugging breakpoint, for development only.
- **void sha3_224_hash (const void *data, size_t data_length, uint8_t hashvalue[28])**
  
  Calculate 224 bit SHA3 digest from data, see also class SHA3_224.
- **void sha3_256_hash (const void *data, size_t data_length, uint8_t hashvalue[32])**
  
  Calculate 256 bit SHA3 digest from data, see also class SHA3_256.
- **void sha3_384_hash (const void *data, size_t data_length, uint8_t hashvalue[48])**
  
  Calculate 384 bit SHA3 digest from data, see also class SHA3_384.
- **void sha3_512_hash (const void *data, size_t data_length, uint8_t hashvalue[64])**
  
  Calculate 512 bit SHA3 digest from data, see also class SHA3_512.
- **void shake128_hash (const void *data, size_t data_length, uint8_t *hashvalues, size_t n)**
  
  Calculate SHA3 extendable output digest for 128 bit security strength, see also class SHAKE128.
- **void shake256_hash (const void *data, size_t data_length, uint8_t *hashvalues, size_t n)**
  
  Calculate SHA3 extendable output digest for 256 bit security strength, see also class SHAKE256.
- **uint64_t random_int64 ()**
  
  Generate a non-deterministic, uniformly distributed 64 bit pseudo-random number.
- **int64_t random_irange (int64_t begin, int64_t end)**
  
  Generate uniformly distributed pseudo-random integer within range.
- **double random_float ()**
  
  Generate uniformly distributed pseudo-random floating point number.
- **double random_frange (double begin, double end)**
  
  Generate uniformly distributed pseudo-random floating point number within a range.
- **uint64_t random_nonce ()**
  
  Provide a unique 64 bit identifier that is not 0, see also random_int64().
- **void random_secret (uint64_t *secret_var)**
  
  Generate a secret non-zero nonce in secret_var, unless it has already been assigned.
- **std::string beastbse_cachedir_create ()**
  
  Create exclusive cache directory for this process' runtime.
- **std::string beastbse_cachedir_current ()**
  
  Retrieve (or create) the temporary cache directory for this runtime.
- **void beastbse_cachedir_cleanup ()**
  
  Clean stale cache directories from past runtimes, may be called from any thread.
- **String string_multiply (const String &s, uint64 count)**
  
  Reproduce a string s for count times.
- **String string_canonify (const String &string, const String &valid_chars, const String &substitute)**
  
  Enforce a canonical charset for a string.
- **bool string_is_canonified (const String &string, const String &valid_chars)**
  
  Check if string_canonify() would modify string.
- **String string_set_a2z ()**
  
  Returns a string containing all of a-z.
- **String string_set_A2Z ()**
  
  Returns a string containing all of A-Z.
- **String string_set_ascii_alnum ()**
  
  Returns a string containing all of 0-9, A-Z and a-z.
- **String string_tolower (const String &str)**
  
  Convert all string characters into Unicode lower case characters.
- **String string_toupper (const String &str)**
  
  Convert all string characters into Unicode upper case characters.
- **String string_totitle (const String &str)**
  
  Convert all string characters into Unicode title characters.
• **String string_capitalize** (const String &str, size_t maxn)
  *Capitalize words, so the first letter is upper case, the rest lower case.*

• **String string_normalize_nfc** (const String &src)
  *Yield normalized composed UTF-8 string.*

• **String string_normalize_nfd** (const String &src)
  *Yield normalized decomposed UTF-8 string.*

• **String string_normalize_nfkc** (const String &src)
  *Formatting stripped normalized composed UTF-8 string.*

• **String string_normalize_nfkd** (const String &src)
  *Formatting stripped normalized decomposed UTF-8 string.*

• **String string_casefold** (const String &src)
  *Yield UTF-8 string useful for case insensitive comparisons.*

• **int string_cmp** (const String &s1, const String &s2)
  *Like strcmp(3) for UTF-8 strings.*

• **int string_casecmp** (const String &s1, const String &s2)
  *Like strcasecmp(3) for UTF-8 strings.*

• **String string_vprintf** (const char *format, va_list vargs)
  *Formatted printing ala vprintf() into a String, using the POSIX/C locale.*

• **String string_locale_vprintf** (const char *format, va_list vargs)
  *Formatted printing like string_vprintf using the current locale.*

• **StringVector string_split** (const String &string, const String &splitter, size_t maxn)
  *Split a string, using splitter as delimiter.*

• **StringVector string_split_any** (const String &string, const String &splitchars, size_t maxn)
  *Split a string, using any of the splitchars as delimiter.*

• **void string_vector_erase_empty** (StringVector &svector)
  *Remove empty elements from a string vector.*

• **void string_vector_lstrip** (StringVector &svector)
  *Left-strip all elements of a string vector, see string_lstrip().*

• **void string_vector_rstrip** (StringVector &svector)
  *Right-strip all elements of a string vector, see string_rstrip().*

• **void string_vector_strip** (StringVector &svector)
  *Strip all elements of a string vector, see string_strip().*

• **String string_join** (const String &junctor, const StringVector &strvec)
  *Join a number of strings.*

• **bool string_to_bool** (const String &string, bool fallback)
  *Interpret a string as boolean value.*

• **String string_from_bool** (bool value)
  *Convert a boolean value into a string.*

• **uint64 string_to_uint** (const String &string, size_t *consumed, uint base)
  *Parse a string into a 64bit unsigned integer, optionally specifying the expected number base.*

• **String string_from_uint** (uint64 value)
  *Convert a 64bit unsigned integer into a string.*

• **bool string_has_int** (const String &string)
  *Checks if a string contains a digit, optionally preceeded by whitespaces.*

• **int64 string_to_int** (const String &string, size_t *consumed, uint base)
  *Parse a string into a 64bit integer, optionally specifying the expected number base.*

• **String string_from_int** (int64 value)
  *Convert a 64bit signed integer into a string.*

• **long double posix_locale_strtold** (const char *nptr, char **endptr)
  *Parse a double from a string ala strtod(), trying locale specific characters and POSIX/C formatting.*

• **long double current_locale_strtold** (const char *nptr, char **endptr)
  *Parse a double from a string ala strtod(), trying locale specific characters and POSIX/C formatting.*
• **double** `string_to_double` (const `String` & `string`)
  Parse a double from a string, trying locale specific characters and POSIX/C formatting.

• **double** `string_to_double` (const `char` * `dblstring`, const `char` ** `endptr`)
  Similar to `string_to_double(const String&)`, but returns the first failing character position in `endptr`.

• `String string_from_float` ( `float` value)
  Convert a double into a string, using the POSIX/C locale.

• `String string_from_double` ( `double` value)
  Convert a double into a string, using the POSIX/C locale.

• `vector< double >` `string_to_double_vector` (const `String` & `string`)
  Parse a string into a list of doubles, expects `';' as delimiter.

• `String string_from_double_vector` (const `vector< double >` & `dvec`, const `String` & `delim`)
  Construct a string out of all double values passed in `dvec`, separated by `delim`.

• `String string_from_errno` ( `int` `errno_val`)
  Returns a String describing the passed in `errno` value, similar to `strerror()`.

• `bool string_is_uuid` (const `String` & `uuid_string`)
  Returns whether `uuid_string` contains a properly formatted UUID string.

• `int string_cmp_uuid` (const `String` & `uuid_string1`, const `String` & `uuid_string2`)
  Returns whether `uuid_string1` compares smaller (-1), equal (0) or greater (+1) to `uuid_string2`.

• `bool stringstartswith` (const `String` & `string`, const `String` & `fragment`)
  Returns whether `string` starts with `fragment`.

• `bool string_endswith` (const `String` & `string`, const `String` & `fragment`)
  Returns whether `string` ends with `fragment`.

• `bool string_match_identifier_tail` (const `String` & `ident`, const `String` & `tail`)
  Variant of `string_match_identifier()` that matches `tail` against `ident` at word boundary.

• `bool string_match_identifier` (const `String` & `ident1`, const `String` & `ident2`)
  Check equality of strings canonicalized to "[0-9a-z_]++.

• `String string_from_pretty_function_name` (const `char` * `cxx_pretty_function`)
  Extract the full function name from `PRETTY_FUNCTION`.

• `String string_from_cescape` (const `String` & `str`)
  Escape text like a C string.

• `String string_from_cquote` (const `String` & `str`)
  Returns a string as C string including double quotes.

• `String string_from_cquote` (const `String` & `input`)
  Parse a possibly quoted C string into regular string.

• `String string_lstrip` (const `String` & `input`)
  Strip whitespaces from the left of a string.

• `String string_rstrip` (const `String` & `input`)
  Strip whitespaces from the right of a string.

• `String string_strip` (const `String` & `input`)
  Strip whitespaces from the left and right of a string.

• `String string_replace` (const `String` & `input`, const `String` & `marker`, const `String` & `replacement`, `size_t` `maxn`)
  Replace substring `marker` in `input` with `replacement`, at most `maxn` times.

• `String string_substitute_char` (const `String` & `input`, const `char` `match`, const `char` `subst`)
  Replace all occurrences of `match` in `input` with `subst`.

• `String string_hexdump` (const void * `addr`, `size_t` `length`, `size_t` `initial_offset`)
  Produce hexdump of a memory region.

• `void memset4` ( `uint32` * `mem`, `uint32` `filler`, `uint` `length`)
  Fill a memory area with a 32-bit quantity.

• `String string_vector_find` (const `StringVector` & `svector`, const `String` & `prefix`, const `String` & `fallback`)
  Search for `prefix` in `svector` and return the matching element.

• `String string_vector_find_value` (const `StringVector` & `svector`, const `String` & `prefix`, const `String` & `fallback`)
  Search for `prefix` in `svector` and return reminder of the matching string.
1.1 Bse Namespace Reference

- **StringVector cstrings_to_vector** (const char *s,...)
  
  Construct a StringVector from a NULL terminated list of string arguments.

- **void string_options_split** (const String &option_string, vector<String> &option_names, vector<String> &option_values, const String &empty_default)
  
  Split an option list string into name/value pairs.

- **String string_option_get** (const String &option_string, const String &option)
  
  Retrieve the option value from an options list separated by ',' or ':'.

- **bool string_option_check** (const String &option_string, const String &option)
  
  Check if an option is set/unset in an options list string.

- **bool text_convert** (const String &to_charset, String &output_string, const String &from_charset, const String &input_string, const String &fallback_charset, const String &output_mark)
  
  Convert a string from one encoding to another.

- **template<typename Type>**
  
  Type string_to_type (const String &string)
  
  Convert a string to a template argument type, such as bool, int, double.

- **template<typename Type>**
  
  String string_from_type (Type value)
  
  Create a string from a templated argument value, such as bool, int, double.

- **template<class... Args>**
  
  String string_locale_format (const char *format, const Args &...args)
  
  Formatted printing ala printf() into a String, using the current locale.

- **size_t utf8len** (const char *str)
  
  Count valid UTF-8 sequences, invalid sequences are counted as Latin-1 characters.

- **size_t utf8len** (const std::string &str)
  
  Count valid UTF-8 sequences, invalid sequences are counted as Latin-1 characters.

- **size_t utf8_to_unicode** (const char *str, uint32_t *codepoints)
  
  Convert valid UTF-8 sequences to Unicode codepoints, invalid sequences are treated as Latin-1 characters.

- **size_t utf8_to_unicode** (const std::string &str, std::vector<uint32_t> &codepoints)
  
  Convert valid UTF-8 sequences to Unicode codepoints, invalid sequences are treated as Latin-1 characters.

- **std::string string_from_unicode** (const std::vector<uint32_t> &codepoints)
  
  Convert codepoints into an UTF-8 string, using the shortest possible encoding.

- **std::string string_from_unicode** (const std::vector<uint32_t> &codepoints)
  
  Convert codepoints into an UTF-8 string, using the shortest possible encoding.

- **constexpr bool unicode_is_valid** (uint32_t u)
  
  Return whether u is an allowed Unicode codepoint within 0x10FFFF and not part of a UTF-16 surrogate pair.

- **constexpr bool unicode_is_assigned** (uint32_t u)
  
  Return whether u matches any of the assigned Unicode planes.

- **constexpr bool unicode_is_noncharacter** (uint32_t u)
  
  Return whether u is one of the 66 Unicode noncharacters.

- **constexpr bool unicode_is_character** (uint32_t u)
  
  Return whether u is not one of the 66 Unicode noncharacters.

- **constexpr bool unicode_is_control_code** (uint32_t u)
  
  Return whether u is one of the 65 Unicode control codes.

- **constexpr bool unicode_is_private** (uint32_t u)
  
  Return whether u is in one of the 3 private use areas of Unicode.
### Variables

- **Const KAMMER_NOTE**
  
  Value represents unparseable/unknown notes.

- **Const KAMMER_FREQ**
  
  Kammer note, representing the kammer frequency's MIDI note value for A' or A4.

- **Const KAMMER_OCTAVE**
  
  Pitch Standard, see also: https://en.wikipedia.org/wiki/A440_(pitch_standard)

- **Const MIN_OCTAVE**
  
  Octave number for MIDI A'.

- **Const MAX_OCTAVE**
  
  Octave of MIN_NOTE.

- **Const MIN_FINE_TUNE**
  
  Octave of MAX_NOTE.

- **uint64_t cached_hash_secret**
  
  Use hash_secret() for access.

### Detailed Description

The **Bse** namespace contains all functions of the synthesis engine.

### Typedef Documentation

```cpp
int16
typedef int16_t Bse::int16
A 16-bit signed integer.
```

```cpp
int32
typedef int32_t Bse::int32
A 32-bit signed integer.
```

```cpp
int64
typedef int64_t Bse::int64
A 64-bit unsigned integer, use PRI+64 in format strings.
```

```cpp
int8
typedef int8_t Bse::int8
An 8-bit signed integer.
```

### String

```cpp
typedef std::string Bse::String
Convenience alias for std::string.
```
StringVector

typedef vector< String > Bse::StringVector
Convenience alias for a std::vector< std::string >.

uint

typedef uint32_t Bse::uint
Provide 'uint' as convenience type.

uint16

typedef uint16_t Bse::uint16
A 16-bit unsigned integer.

uint32

typedef uint32_t Bse::uint32
A 32-bit unsigned integer.

uint64

typedef uint64_t Bse::uint64
A 64-bit unsigned integer, use PRI+64 in format strings.

uint8

typedef uint8_t Bse::uint8
An 8-bit unsigned integer.

unichar

typedef uint32_t Bse::unichar
A 32-bit unsigned integer used for Unicode characters.

Enumeration Type Documentation

ModuleFlag

enum Bse::ModuleFlag [strong]

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NORMAL</td>
<td>Normal flag</td>
</tr>
<tr>
<td>CHEAP</td>
<td>Very short or NOP as process() function.</td>
</tr>
<tr>
<td>EXPENSIVE</td>
<td>Indicate lengthy process() function.</td>
</tr>
<tr>
<td>VIRTUAL</td>
<td>Flag used internally.</td>
</tr>
</tbody>
</table>
MonitorField

```cpp
enum Bse::MonitorField
Offsets for signal monitoring fields in bytes, field type and size is used as prefix.
```

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F64_GENERATION</td>
<td>Generation counter, updated on every modification.</td>
</tr>
<tr>
<td>F32_MIN</td>
<td>Minimum value of the last frame.</td>
</tr>
<tr>
<td>F32_MAX</td>
<td>Maximum value of the last frame.</td>
</tr>
<tr>
<td>F32_DB_SPL</td>
<td>Sound pressure level in dB SPL of the last frame.</td>
</tr>
<tr>
<td>F32_DB_TIP</td>
<td>Maximum recent dB SPL.</td>
</tr>
<tr>
<td>END_BYTE</td>
<td>Total length of all MonitorField values in bytes.</td>
</tr>
</tbody>
</table>

ProjectState

```cpp
enum Bse::ProjectState
Enumeration describing the current activation and playback state of a project.
```

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INACTIVE</td>
<td>The project is not yet hooked to the sound engine.</td>
</tr>
<tr>
<td>ACTIVE</td>
<td>The sound engine is activated (runing) for this project.</td>
</tr>
<tr>
<td>PLAYING</td>
<td>The project is active and the sequencer is running.</td>
</tr>
</tbody>
</table>

SongTelemetry

```cpp
enum Bse::SongTelemetry
Offsets for signal monitoring fields in bytes, field type and size is used as prefix.
```

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>I32_TICK_POINTER</td>
<td>Current song position pointer.</td>
</tr>
<tr>
<td>BYTECOUNT</td>
<td>Total length of all fields.</td>
</tr>
</tbody>
</table>

UserMessageType

```cpp
enum Bse::UserMessageType
```

<table>
<thead>
<tr>
<th>Enumerator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR</td>
<td>Indicate a message about an error condition.</td>
</tr>
<tr>
<td>WARNING</td>
<td>Indicate a message about a possibly harmful condition.</td>
</tr>
<tr>
<td>INFO</td>
<td>Indicate an informational message.</td>
</tr>
<tr>
<td>DEBUG</td>
<td>Indicate a debugging message (usually insignificant).</td>
</tr>
</tbody>
</table>
### Function Documentation

_.O_ [1/4]

```cpp
const char* Bse::_ (const char* string)
```

Translate message strings in the BEAST/BSE text domain.

_.O_ [2/4]

```cpp
std::string Bse::_ (const std::string & string)
```

Translate message strings in the BEAST/BSE text domain.

_.O_ [3/4]

```cpp
const char* Bse::_ (const char* string, const char* plural, int64_t n)
```

Translate message strings in the BEAST/BSE text domain, use forms if != 1.

_.O_ [4/4]

```cpp
std::string Bse::_ (const std::string & string, const std::string & plural, int64_t n)
```

Translate message strings in the BEAST/BSE text domain, use forms if != 1.

### aligned_alloc()

```cpp
void * Bse::aligned_alloc (size_t total_size, size_t alignment, uint8 ** free_pointer)
```

Allocate a block of memory aligned to at least `alignment` bytes.

### aligned_free()

```cpp
void Bse::aligned_free (uint8 ** free_pointer)
```

Release a block of memory allocated through `aligned_malloc()`.

### allocate_aligned_block()

```cpp
AlignedBlock Bse::allocate_aligned_block (uint32 mem_id, uint32 length)
```

Create a memory block from memory area `mem_id`, if 0, uses the internal cache-line aligned pool.
application_name()

String Bse::application_name ( )
Retrieve the localized program name intended for user display.

application_name_init()

void Bse::application_name_init ( )
Set the application_name to a name other than program_alias if desired.

beastbse_cachedir_cleanup()

void Bse::beastbse_cachedir_cleanup ( )
Clean stale cache directories from past runtimes, may be called from any thread.

beastbse_cachedir_create()

std::string Bse::beastbse_cachedir_create ( )
Create exclusive cache directory for this process' runtime.

beastbse_cachedir_current()

std::string Bse::beastbse_cachedir_current ( )
Retrieve (or create) the temporary cache directory for this runtime.

binary_lookup()

template<typename RandIter , class Cmp , typename Arg >
RandIter Bse::binary_lookup ( 
    RandIter begin,
    RandIter end,
    Cmp cmp_elements,
    const Arg & arg ) [inline]
Perform binary lookup and yield exact match or end.
The arguments [ begin, end ] denote the range used for the lookup, arg is passed along with the current element to the cmp_elements function.

binary_lookup_insertion_pos()

template<typename RandIter , class Cmp , typename Arg >
std::pair<RandIter,bool> Bse::binary_lookup_insertion_pos ( 
    RandIter begin,
    RandIter end,
    Cmp cmp_elements,
    const Arg & arg ) [inline]
Perform a binary lookup to find the insertion position for a new element.
Return (end,false) for end-begin == 0, or return (position,true) for exact match, otherwise return (position,false) where position indicates the location for the key to be inserted (and may equal end).
**binary_lookup_sibling()**

```cpp
template<typename RandIter , class Cmp , typename Arg >
RandIter Bse::binary_lookup_sibling (  
    RandIter begin,  
    RandIter end,  
    Cmp cmp_elements,  
    const Arg & arg ) [inline]
```

Perform a binary lookup to yield exact or nearest match.
return end for end-begin == 0, otherwise return the exact match element, or, if there's no such element, return the element last visited, which is pretty close to an exact match (will be one off into either direction).

**breakpoint()**

```cpp
void Bse::breakpoint ( ) [inline]
```

Cause a debugging breakpoint, for development only.

**collect_runtime_entropy()**

```cpp
void Bse::collect_runtime_entropy (  
    uint64 * data,  
    size_t n )
```

To provide good quality random number seeds, this function gathers entropy from a variety of process specific sources.
Collect entropy from the current process, usually quicker than `collect_system_entropy()`.
Under Linux, this includes the CPU counters, clocks and random devices. In combination with well established techniques like syscall timings (see Entropics13 Swanson (2013)) and a SHA3 algorithm derived random number generator for the mixing, the entropy collection is designed to be fast and good enough for all non-cryptographic uses. On an Intel Core i7, this function takes around 25µs.

**collect_system_entropy()**

```cpp
void Bse::collect_system_entropy (  
    uint64 * data,  
    size_t n )
```

This function adds to `collect_runtime_entropy()` by collecting entropy from additional but potentially slower system sources, such as interrupt counters, disk + network statistics, system load, execution + pipelining + scheduling latencies and device MACs.
Collect entropy from system devices, like interrupt counters, clocks and random devices.
The function is designed to yield random number seeds good enough to generate cryptographic tokens like session keys. On an Intel Core i7, this function takes around 2ms, so it's roughly 80 times slower than `collect_runtime_entropy()`.

**constexpr_equals()**

```cpp
constexpr bool Bse::constexpr_equals (  
    const char * a,  
    const char * b,  
    size_t n ) [inline]
```

Test string equality at compile time.

**copy_reordered()**

```cpp
template<class InputIterator , class OutputIterator >
OutputIterator Bse::copy_reordered (  
    InputIterator const unordered_first,  
    InputIterator const unordered_end,  
    InputIterator const ordered_first,
```

Bse 0.15.0 November 2019 Beast Sound Engine
Copy \texttt{unordered\_first} .. \texttt{unordered\_end} into \texttt{output\_iterator} in the order given by \texttt{ordered\_first} .. \texttt{ordered\_end}.

\texttt{cpu\_arch()}

\begin{verbatim}
std::string Bse::cpu\_arch ( )
\end{verbatim}
Retrieve string identifying the CPU architecture.

\texttt{cpu\_info()}

\begin{verbatim}
std::string Bse::cpu\_info ( )
\end{verbatim}
The returned string contains: number of online CPUs, a string describing the CPU architecture, the vendor and finally a number of flag words describing CPU features plus a trailing space.
Retrieve string identifying the runtime CPU type.
This allows checks for CPU features via a simple string search for \texttt{"FEATURE"}.

Returns
Example: 
\begin{verbatim}
"4 AMD64 GenuineIntel FPU TSC HTT CMPXCHG16B MMX MMXEXT SSESYS SSE SSE2 SSE3 SSSE3 SSE4.1 SSE4.2"
\end{verbatim}

\texttt{create\_memory\_area()}

\begin{verbatim}
MemoryArea Bse::create\_memory\_area ( 
  uint32 mem\_size, 
  uint32 alignment )
\end{verbatim}
Create isolated memory area, the MemoryArea.mem\_id can be used for \texttt{allocate\_aligned\_block()}. 

\texttt{cstrings\_to\_vector()}

\begin{verbatim}
StringVector Bse::cstrings\_to\_vector ( 
  const char * s, 
  ... )
\end{verbatim}
Construct a StringVector from a NULL terminated list of string arguments.

\texttt{current\_locale\_strtold()}

\begin{verbatim}
long double Bse::current\_locale\_strtold ( 
  const char * nptr, 
  char ** endptr )
\end{verbatim}
Parse a double from a string ala \texttt{strtold()}, trying locale specific characters and POSIX/C formatting.

\texttt{debug()}

\begin{verbatim}
template<class ... Args>
void Bse::debug ( 
  const char * cond, 
  const char * format, 
  const Args &... args ) [inline]
\end{verbatim}
Issue a printf-like debugging message if \texttt{cond} is enabled by $BSE\_DEBUG$. 
debug_enabled()

bool Bse::debug_enabled ( ) [inline]
Check if any kind of debugging is enabled by $BSE_DEBUG.

d-debug_key_enabled() [1/2]

bool Bse::debug_key_enabled ( 
const char * conditional )
Check if conditional is enabled by $BSE_DEBUG.

d-debug_key_enabled() [2/2]

bool Bse::debug_key_enabled ( 
const :: std::string & conditional )
Check if conditional is enabled by $BSE_DEBUG.

d-debug_key_value()

std::string Bse::debug_key_value ( 
const char * conditional )
Retrieve the value assigned to debug key conditional in $BSE_DEBUG.

d-debug_message()

template<class ... Args>
void Bse::debug_message ( 
const char * file, 
int line, 
const char * func, 
const char * cond, 
const char * format, 
const Args &... args ) [inline]
Issue a printf-like debugging message if cond is enabled by $BSE_DEBUG.

d-delete_inplace()

template<class Type >
void Bse::delete_inplace ( 
Type & typemem ) [inline]
Call inplace delete operator by automatically inferring the Type.

d-diag_abort_hook()

void Bse::diag_abort_hook ( 
const std::function< void(const :: std::string &)> & hook )
Call hook for fatal_error() and diag_failed_assert().

d-exec_handler_clear()

bool Bse::exec_handler_clear ( 
uint id )
Remove a function previously added with exec_now() or exec_timeout()
exec_now() [1/2]

```cpp
uint Bse::exec_now ( const std::function<bool()> & function )
```

Run `function` immediately with the next event loop iteration, return `true` to keep alive.

---

exec_now() [2/2]

```cpp
uint Bse::exec_now ( const std::function<void()> & function )
```

Run `function` immediately with the next event loop iteration.

---

exec_timeout() [1/2]

```cpp
uint Bse::exec_timeout ( const std::function<bool()> & function, uint delay_ms )
```

Run `function` after `delay_ms` milliseconds have passed, return `true` to keep alive.

---

exec_timeout() [2/2]

```cpp
uint Bse::exec_timeout ( const std::function<void()> & function, uint delay_ms )
```

Run `function` after `delay_ms` milliseconds have passed.

---

executable_name()

```cpp
std::string Bse::executable_name ( )
```

Retrieve the name part of `executable_path()`.

---

executable_path()

```cpp
std::string Bse::executable_path ( )
```

Retrieve the path to the currently running executable.

---

fatal_error() [1/2]

```cpp
template<class ... Args>
String String void Bse::fatal_error ( const char * format, const Args &... args )
```

Issue a printf-like message and abort the program, this function will not return.

Avoid using this in library code, aborting may take precious user data with it, library code should instead use `warning()`, `info()` or `assert_return()`.

---

fatal_error() [2/2]

```cpp
template<class ... Args>
void Bse::fatal_error ( const char * format, const Args &... args )
```

Issue a printf-like message and abort the program, this function will not return.

Avoid using this in library code, aborting may take precious user data with it, library code should instead use `warning()`, `info()` or `assert_return()`.
feature_check()

bool Bse::feature_check(const char * feature)
Check if feature is enabled via $BSE_FEATURE.

feature_toggle_bool()

bool Bse::feature_toggle_bool(const char * config, const char * feature)
Check for feature in config, if feature is empty, checks for any feature.

feature_toggle_find()

String Bse::feature_toggle_find(const String & config, const String & feature, const String & fallback)
Find feature in config, return its value or fallback.

find_memory_area()

MemoryArea Bse::find_memory_area(uint32 mem_id)
Lookup a previously created memory area.

fmsb()

int Bse::fmsb(uint64 val)
The fmsb() function returns the position of the most significant bit set in the word val.
Find most significant bit set in a word.
The least significant bit is position 1 and the most significant position is, for example, 32 or 64.

Returns
The position of the most significant bit set is returned, or 0 if no bits were set.

icon_from_pixstream()

Icon Bse::icon_from_pixstream(const uint8 * pixstream)
Create a Bse::Icon from a GdkPixbuf pixstream.

icon_sanitize()

bool Bse::icon_sanitize(Icon * icon)
Ensure consistency of the icon fields.
info()

template<class ... Args>
void Bse::info (const char * format, const Args & ... args)

Issue an informative printf-like message.

init_async()

void Bse::init_async (int * argc, char ** argv, const char * app_name, const StringVector & args)

Initialize and start BSE. Initialize the BSE library and start the main BSE thread. Arguments specific to BSE are removed from argc / argv.

init_glue_context()

SfiGlueContext * Bse::init_glue_context (const gchar * client, const std::function<void()> & caller_wakeup)

Create SFI glue layer context. Create and push an SFI glue layer context for the calling thread, to enable communications with the main BSE thread library.

init_needed()

bool Bse::init_needed ()

Check whether init_async() still needs to be called.

init_server_instance()

ServerHandle Bse::init_server_instance ()

Retrieve a handle for the Bse::Server instance managing the Bse thread.

memset4()

void Bse::memset4 (uint32 * mem, uint32 filler, uint length)

Fill a memory area with a 32-bit quantity.

monotonic_counter()

uint64 Bse::monotonic_counter ()

A monotonically increasing counter, increments are atomic and visible in all threads.
new_inplace()

```cpp
template<class Type , class ... Ts>
void Bse::new_inplace ( 
    Type & typemem,
    Ts &&... args ) [inline]
```

Call inplace new operator by automatically inferring the Type.

posix_locale_strtold()

```cpp
long double Bse::posix_locale_strtold ( 
    const char * nptr,
    char ** endptr )
```

Parse a double from a string ala `strtod()`, trying locale specific characters and POSIX/C formatting.

print_backtrace()

```cpp
bool Bse::print_backtrace ( 
    const char * file,
    int line,
    const char * func ) [inline]
```

Print a C++ backtrace to stdout/stderr.

printerr()

```cpp
template<class... Args>
void Bse::printerr ( 
    const char * format,
    const Args &... args )
```

Print a message on stderr (and flush stderr) ala `printf()`, using the POSIX/C locale.

printout()

```cpp
template<class... Args>
void Bse::printout ( 
    const char * format,
    const Args &... args )
```

Print a message on stdout (and flush stdout) ala `printf()`, using the POSIX/C locale.

program_alias()

```cpp
String Bse::program_alias ( )
```

Retrieve the program name as used for logging or debug messages.

program_alias_init()

```cpp
void Bse::program_alias_init ( 
    String customname )
```

Set program_alias to a non-localized alias other than program_argv0 if desired.
1.1 Bse Namespace Reference

program_cwd()

```cpp
String Bse::program_cwd ( )
```
The current working directory during startup.

random_float()

```cpp
double Bse::random_float ( )
```
Generate uniformly distributed pseudo-random floating point number. This function generates a pseudo-random number like `random_int64()`, constrained to the range: 0.0 ≤ number < 1.0.

random_frange()

```cpp
double Bse::random_frange ( double begin, double end )
```
Generate uniformly distributed pseudo-random floating point number within a range. This function generates a pseudo-random number like `random_float()`, constrained to the range: begin ≤ number < end.

random_int64()

```cpp
uint64_t Bse::random_int64 ( )
```
Generate a non-deterministic, uniformly distributed 64 bit pseudo-random number. This function generates pseudo-random numbers using the system state as entropy and class `KeccakRng` for the mixing. No seeding is required.

random_irange()

```cpp
int64_t Bse::random_irange ( int64_t begin, int64_t end )
```
Generate uniformly distributed pseudo-random integer within range. This function generates a pseudo-random number like `random_int64()`, constrained to the range: begin ≤ number < end.

random_nonce()

```cpp
uint64_t Bse::random_nonce ( )
```
Provide a unique 64 bit identifier that is not 0, see also `random_int64()`.

random_secret()

```cpp
void Bse::random_secret ( uint64_t *secret_var )
```
Generate a secret non-zero nonce in `secret_var`, unless it has already been assigned.

release_aligned_block()

```cpp
void Bse::release_aligned_block ( const AlignedBlock & am )
```
Realease a previously allocated block.
runpath()

```cpp
std::string Bse::runpath (RPath rpath)
```

Retrieve various resource paths at runtime.

---

sha3_224_hash()

```cpp
void Bse::sha3_224_hash (const void ∗data, size_t data_length, uint8_t hashvalue[28])
```

Calculate 224 bit SHA3 digest from data, see also class SHA3_224.

---

sha3_256_hash()

```cpp
void Bse::sha3_256_hash (const void ∗data, size_t data_length, uint8_t hashvalue[32])
```

Calculate 256 bit SHA3 digest from data, see also class SHA3_256.

---

sha3_384_hash()

```cpp
void Bse::sha3_384_hash (const void ∗data, size_t data_length, uint8_t hashvalue[48])
```

Calculate 384 bit SHA3 digest from data, see also class SHA3_384.

---

sha3_512_hash()

```cpp
void Bse::sha3_512_hash (const void ∗data, size_t data_length, uint8_t hashvalue[64])
```

Calculate 512 bit SHA3 digest from data, see also class SHA3_512.

---

shake128_hash()

```cpp
void Bse::shake128_hash (const void ∗data, size_t data_length, uint8_t ∗hashvalues, size_t n)
```

Calculate SHA3 extendable output digest for 128 bit security strength, see also class SHAKE128.

---

shake256_hash()

```cpp
void Bse::shake256_hash (const void ∗data, size_t data_length, uint8_t ∗hashvalues, size_t n)
```
Calculate SHA3 extendable output digest for 256 bit security strength, see also class SHAKE256.

**shared_ptr_cast** [1/4]

```cpp
template<class Target, class Source>
std::shared_ptr<typename std::remove_pointer<Target>::type> Bse::shared_ptr_cast (
    Source * object )
```

Shorthand for std::dynamic_pointer_cast<(shared_from_this()).

A `shared_ptr_cast()` takes a `std::shared_ptr` or a pointer to an `object` that supports `std::enable_shared_from_this()`. Using `std::dynamic_pointer_cast()`, the `shared_ptr` passed in (or retrieved via calling `shared_from_this()`) is cast into a `std::shared_ptr<Target>`, possibly resulting in an empty (`N-1ULL`) `std::shared_ptr` if the underlying `dynamic_cast()` was not successful or if a NULL `object` was passed in. Note that `shared_from_this()` can throw a `std::bad_weak_ptr` exception if the object has no associated `std::shared_ptr` (usually during ctor and dtor), in which case the exception will also be thrown from `shared_ptr_cast<Target>()`. However a `shared_ptr_cast<Target>()` call will not throw and yield an empty (NULL) `std::shared_ptr<Target>`. This is analogous to `dynamic_cast<T&>` which throws, versus `dynamic_cast<T*>` which yields NULL.

Returns

A `std::shared_ptr<Target>` storing a pointer to `object` or NULL.

Exceptions

| `std::bad_weak_ptr` | if `shared_from_this()` throws, unless the `Target*` form is used. |

**shared_ptr_cast** [2/4]

```cpp
template<class Target, class Source>
const std::shared_ptr<typename std::remove_pointer<Target>::type> Bse::shared_ptr_cast ( 
    const Source * object )
```

See `shared_ptr_cast(Source*)`.

**shared_ptr_cast** [3/4]

```cpp
template<class Target, class Source>
std::shared_ptr<typename std::remove_pointer<Target>::type> Bse::shared_ptr_cast ( 
    std::shared_ptr< Source > & sptr )
```

See `shared_ptr_cast(Source*)`.

**shared_ptr_cast** [4/4]

```cpp
template<class Target, class Source>
const std::shared_ptr<typename std::remove_pointer<Target>::type> Bse::shared_ptr_cast ( 
    const std::shared_ptr< Source > & sptr )
```

See `shared_ptr_cast(Source*)`.

**string_canonify()**

```cpp
String Bse::string_canonify ( 
    const String & string, 
    const String & valid_chars, 
    const String & substitute )
```
Enforce a canonical charset for a string. Convert all chars in string that are not listed as valid_chars with substitute.

**string_capitalize()**

```cpp
String Bse::string_capitalize ( const String & str, size_t maxn )
```

Capitalize words, so the first letter is upper case, the rest lower case.

**string_casecmp()**

```cpp
int Bse::string_casecmp ( const String & s1, const String & s2 )
```

Like strcasecmp(3) for UTF-8 strings.

**string_casefold()**

```cpp
String Bse::string_casefold ( const String & src )
```

Yield UTF-8 string useful for case insensitive comparisons.

**string_cmp()**

```cpp
int Bse::string_cmp ( const String & s1, const String & s2 )
```

Like strcmp(3) for UTF-8 strings.

**string_cmp_uuid()**

```cpp
int Bse::string_cmp_uuid ( const String & uuid_string1, const String & uuid_string2 )
```

Returns whether uuid_string1 compares smaller (-1), equal (0) or greater (+1) to uuid_string2.

**string_endswith()**

```cpp
bool Bse::string_endswith ( const String & string, const String & fragment )
```

Returns whether string ends with fragment.

**string_format()**

```cpp
template<class... Args>
String Bse::string_format ( const char ∗ format, const Args &... args )
```

Formatted printing ala printf() into a String, using the POSIX/C locale.
1.1 Bse Namespace Reference

string_from_bool()

String Bse::string_from_bool (bool value)
Convert a boolean value into a string.

string_from_cquote()

String Bse::string_from_cquote (const String & input)
Parse a possibly quoted C string into regular string.

string_from_double()

String Bse::string_from_double (double value)
Convert a double into a string, using the POSIX/C locale.

string_from_double_vector()

String Bse::string_from_double_vector (const vector<double> & dvec, const String & delim)
Construct a string out of all double values passed in dvec, separated by delim.

string_from_errno()

String Bse::string_from_errno (int errno_val)
Returns a String describing the passed in errno value, similar to strerror().

string_from_float()

String Bse::string_from_float (float value)
Convert a float into a string, using the POSIX/C locale.

string_from_int()

String Bse::string_from_int (int64 value)
Convert a 64bit signed integer into a string.

string_from_pretty_function_name()

String Bse::string_from_pretty_function_name (const char * cxx_pretty_function)
Extract the full function name from PRETTY_FUNCTION.
See also BSE_SIMPLE_FUNCTION.
string_from_type()

template<typename Type >
String Bse::string_from_type ( 
    Type value )
Create a string from a templated argument value, such as bool, int, double.

string_from_uint()

String Bse::string_from_uint ( 
    uint64 value )
Convert a 64bit unsigned integer into a string.

string_from_unicode() [1/2]

std::string Bse::string_from_unicode ( 
    const uint32_t * codepoints, 
    size_t n_codepoints )
Convert codepoints into an UTF-8 string, using the shortest possible encoding.

string_from_unicode() [2/2]

std::string Bse::string_from_unicode ( 
    const std::vector< uint32_t > & codepoints )
Convert codepoints into an UTF-8 string, using the shortest possible encoding.

string_has_int()

bool Bse::string_has_int ( 
    const String & string )
Checks if a string contains a digit, optionally preceeded by whitespaces.

string_hexdump()

String Bse::string_hexdump ( 
    const void * addr, 
    size_t length, 
    size_t initial_offset )
Produce hexdump of a memory region. 
Each output line consists of its hexadecimal offset, 16 hexadecimal bytes and the ASCII representation of the same 16 bytes.

string_is_canonified()

bool Bse::string_is_canonified ( 
    const String & string, 
    const String & valid_chars )
Check if string_canonify() would modify string.

string_is_uuid()

bool Bse::string_is_uuid ( 
    const String & uuid_string )
Returns whether uuid_string contains a properly formatted UUID string.
string_join()

```cpp
String Bse::string_join (
    const String & juncor,
    const StringVector & strvec)
```

Join a number of strings.
Join a string vector into a single string, using `juncor` inbetween each pair of strings.

string/locale_format()

```cpp
String Bse::string_locale_format (  
    const char * format,  
    const Args &... args)
```

Formatted printing ala `printf()` into a String, using the current locale.

string/locale_vprintf()

```cpp
String Bse::string_locale_vprintf (  
    const char * format,  
    va_list vargs)
```

Formatted printing like `string_vprintf` using the current locale.

string_lstrip()

```cpp
String Bse::string_lstrip (  
    const String & input)
```

Strip whitespaces from the left of a string.

string_match_identifier()

```cpp
bool Bse::string_match_identifier (   
    const String & ident1,  
    const String & ident2)
```

Check equality of strings canonicalized to ”[0-9a-z] + “.

string_match_identifier_tail()

```cpp
bool Bse::string_match_identifier_tail (  
    const String & ident,  
    const String & tail)
```

Variant of `string_match_identifier()` that matches `tail` against `ident` at word boundary.

string_multiply()

```cpp
String Bse::string_multiply (  
    const String & s,  
    const String & t)
```

Reproduce a string `s` for `count` times.
string_normalize_nfc()

String Bse::string_normalize_nfc (const String & src)

Yield normalized composed UTF-8 string.

string_normalize_nfd()

String Bse::string_normalize_nfd (const String & src)

Yield normalized decomposed UTF-8 string.

string_normalize_nfkc()

String Bse::string_normalize_nfkc (const String & src)

Formatting stripped normalized composed UTF-8 string.

string_normalize_nfkd()

String Bse::string_normalize_nfkd (const String & src)

Formatting stripped normalized decomposed UTF-8 string.

string_option_check()

bool Bse::string_option_check (const String & option_string, const String & option)

Check if an option is set/unset in an options list string.

string_option_get()

String Bse::string_option_get (const String & option_string, const String & option)

Retrieve the option value from an options list separated by ',' or ';'.

string_options_split()

void Bse::string_options_split (const String & option_string, vector<String> & option_names, vector<String> & option_values, const String & empty_default)

Split an option list string into name/value pairs.

string_replace()

String Bse::string_replace (const String & input, const String & marker, const String & replacement, size_t maxn)
Replace substring *marker* in *input* with *replacement*, at most *maxn* times.

**string_rstrip()**

```cpp
String Bse::string_rstrip (const String & input)
```
Strip whitespaces from the right of a string.

**string_set_a2z()**

```cpp
String Bse::string_set_a2z ( )
```
Returns a string containing all of a-z.

**string_set_A2Z()**

```cpp
String Bse::string_set_A2Z ( )
```
Returns a string containing all of A-Z.

**string_set_ascii_alnum()**

```cpp
String Bse::string_set_ascii_alnum ( )
```
Returns a string containing all of 0-9, A-Z and a-z.

**string_split()**

```cpp
StringVector Bse::string_split (const String & string, const String & splitter, size_t maxn)
```
Split a string, using *splitter* as delimiter. Passing "" as *splitter* will split the string at whitespace positions.

**string_split_any()**

```cpp
StringVector Bse::string_split_any (const String & string, const String & splitchars, size_t maxn)
```
Split a string, using any of the *splitchars* as delimiter. Passing "" as *splitter* will split the string between all position.

**string_startswith()**

```cpp
bool Bse::string_startswith (const String & string, const String & fragment)
```
Returns whether *string* starts with *fragment*.

**string_strip()**

```cpp
String Bse::string_strip (const String & input)
```
Strip whitespaces from the left and right of a string.
string_substitute_char()

```cpp
String Bse::string_substitute_char (  
    const String & input,  
    const char match,  
    const char subst )
```

Replace all occurrences of `match` in `input` with `subst`.

string_to_bool()

```cpp
bool Bse::string_to_bool (  
    const String & string,  
    bool fallback )
```

Interpret a string as boolean value.
Interpret the string as number, "ON"/"OFF" or distinguish "false"/"true" or "yes"/"no" by starting letter. For empty strings, `fallback` is returned.

string_to_cescape()

```cpp
String Bse::string_to_cescape (  
    const String & str )
```

Escape text like a C string. Returns a string that escapes all characters with a backslash `\` that need escaping in C language string syntax.

string_to_cquote()

```cpp
String Bse::string_to_cquote (  
    const String & str )
```

Returns a string as C string including double quotes.

string_to_double() [1/2]

```cpp
double Bse::string_to_double (  
    const String & string )
```

Parse a double from a string, trying locale specific characters and POSIX/C formatting.

string_to_double() [2/2]

```cpp
double Bse::string_to_double (  
    const char * dblstring,  
    const char ** endptr )
```

Similar to `string_to_double(const String&)`, but returns the first failing character position in `endptr`.

string_to_double_vector()

```cpp
vector<double> Bse::string_to_double_vector (  
    const String & string )
```

Parse a string into a list of doubles, expects ';' as delimiter.
string_to_int()

```cpp
int64 Bse::string_to_int (const String & string, size_t * consumed, uint base)
```

Parse a string into a 64bit integer, optionally specifying the expected number base.

string_to_type()

```cpp
template<typename Type>
Type Bse::string_to_type (const String & string)
```

Convert a string to template argument type, such as bool, int, double.

string_to_uint()

```cpp
uint64 Bse::string_to_uint (const String & string, size_t * consumed, uint base)
```

Parse a string into a 64bit unsigned integer, optionally specifying the expected number base.

string_tolower()

```cpp
String Bse::string_tolower (const String & str)
```

Convert all string characters into Unicode lower case characters.

string_totitle()

```cpp
String Bse::string_totitle (const String & str)
```

Convert all string characters into Unicode title characters.

string_toupper()

```cpp
String Bse::string_toupper (const String & str)
```

Convert all string characters into Unicode upper case characters.

string_vector_erase_empty()

```cpp
void Bse::string_vector_erase_empty (StringVector & svector)
```

Remove empty elements from a string vector.

string_vector_find()

```cpp
String Bse::string_vector_find (const StringVector & svector, const String & prefix, const String & fallback)
```

Search for prefix in svector and return the matching element.
If multiple matches are possible, the last one is returned. Returns

    fallback if no match was found.

**string_vector_find_value()**

```cpp
String Bse::string_vector_find_value (  
    const StringVector & svector,  
    const String & prefix,  
    const String & fallback  
)
```

Search for `prefix` in `svector` and return reminder of the matching string. If multiple matches are possible, the last one is returned. Returns

    fallback if no match was found.

**string_vector_lstrip()**

```cpp
void Bse::string_vector_lstrip (  
    StringVector & svector  
)
```

Left-strip all elements of a string vector, see `string_lstrip()`.

**string_vector_rstrip()**

```cpp
void Bse::string_vector_rstrip (  
    StringVector & svector  
)
```

Right-strip all elements of a string vector, see `string_rstrip()`.

**string_vector_strip()**

```cpp
void Bse::string_vector_strip (  
    StringVector & svector  
)
```

Strip all elements of a string vector, see `string_strip()`.

**string_vprintf()**

```cpp
String String Bse::string_vprintf (  
    const char * format,  
    va_list vargs  
)
```

Formatted printing ala `vprintf()` into a String, using the POSIX/C locale.

**text_convert()**

```cpp
bool Bse::text_convert (  
    const String & to_charset,  
    String & output_string,  
    const String & from_charset,  
    const String & input_string,  
    const String & fallback_charset,  
    const String & output_mark  
)
```

Convert a string from one encoding to another.
Convert \textit{input string} from encoding \textit{from charset} to \textit{to charset}, returning \textit{output string}. Interpret unknown characters according to \textit{fallback charset}. Use \textit{output mark} in place of unconvertible characters. Returns whether the conversion was successful.

\textbf{timestamp\_benchmark()}

\texttt{uint64 Bse::timestamp\_benchmark ( )}

Returns benchmark timestamp in nano-seconds, clock starts around program startup.

\textbf{timestamp\_format()}

\texttt{String Bse::timestamp\_format ( uint64 stamp, uint maxlength )}

Convert \textit{stamp} into a string, adding \textmu second fractions if space permits.

\textbf{timestamp\_realtime()}

\texttt{uint64 Bse::timestamp\_realtime ( )}

Return the current time as uint64 in \textmu seconds.

\textbf{timestamp\_resolution()}

\texttt{uint64 Bse::timestamp\_resolution ( )}

Provide resolution of \textbf{timestamp\_benchmark()} in nano-seconds.

\textbf{timestamp\_startup()}

\texttt{uint64 Bse::timestamp\_startup ( )}

Provides the \textbf{timestamp\_realtime()} value from program startup.

\textbf{unicode\_is\_assigned()}

\texttt{constexpr bool Bse::unicode\_is\_assigned ( uint32\_t u ) [inline]}

Return whether \textit{u} matches any of the assigned Unicode planes.

\textbf{unicode\_is\_character()}

\texttt{constexpr bool Bse::unicode\_is\_character ( uint32\_t u ) [inline]}

Return whether \textit{u} is not one of the 66 Unicode noncharacters.

\textbf{unicode\_is\_control\_code()}

\texttt{constexpr bool Bse::unicode\_is\_control\_code ( uint32\_t u ) [inline]}

Return whether \textit{u} is one of the 65 Unicode control codes.
**unicode_is_noncharacter()**

```cpp
constexpr bool Bse::unicode_is_noncharacter ( uint32_t u ) [inline]
```

Return whether \( u \) is one of the 66 Unicode noncharacters.

**unicode_is_private()**

```cpp
constexpr bool Bse::unicode_is_private ( uint32_t u ) [inline]
```

Return whether \( u \) is in one of the 3 private use areas of Unicode.

**unicode_is_valid()**

```cpp
constexpr bool Bse::unicode_is_valid ( uint32_t u ) [inline]
```

Return whether \( u \) is an allowed Unicode codepoint within 0x10FFFF and not part of a UTF-16 surrogate pair.

**url_show()**

```cpp
bool Bse::url_show ( const char * url )
```

Find a suitable WWW user agent (taking user configurations into account) and start it to display \( url \).
Display \( url \) via a suitable WWW user agent.
Several user agents are tried before giving up.

Returns

- **True** if a user agent could be launched successfully.

**utf8_to_unicode() [1/2]**

```cpp
size_t Bse::utf8_to_unicode ( const char * str, uint32_t * codepoints )
```

Convert valid UTF-8 sequences to Unicode codepoints, invalid sequences are treated as Latin-1 characters.
The array `codepoints` must be able to hold at least as many elements as are characters stored in `str`. Returns the number of codepoints stored in `codepoints`.

**utf8_to_unicode() [2/2]**

```cpp
size_t Bse::utf8_to_unicode ( const std::string & str, std::vector< uint32_t > & codepoints )
```

Convert valid UTF-8 sequences to Unicode codepoints, invalid sequences are treated as Latin-1 characters.
Returns the number of codepoints newly stored in `codepoints`.

**utf8len() [1/2]**

```cpp
size_t Bse::utf8len ( const char * str )
```

Count valid UTF-8 sequences, invalid sequences are counted as Latin-1 characters.
utf8len() [2/2]

```cpp
size_t Bse::utf8len (const std::string & str )
```

Count valid UTF-8 sequences, invalid sequences are counted as Latin-1 characters.

vector_erase_element()

```cpp
template< class V >
bool Bse::vector_erase_element (V & v,
    const typename V::value_type & value )
```

Erase element `value` from `std::vector v` if it is present.

vector_erase_iface()

```cpp
template< class V , class O >
bool Bse::vector_erase_iface (V & v,
    O * value )
```

Erase element `value` from `std::vector v` if it matches a vector elements `iface()`.

version()

```cpp
std::string Bse::version ()
```

Provide a string containing the BSE library build.

warning()

```cpp
template< class ... Args>
void Bse::warning (const char * format,
    const Args &... args )
```

Issue a printf-like warning message.

zintern_decompress()

```cpp
uint8 * Bse::zintern_decompress (unsigned int decompressed_size,
    const unsigned char * cdata,
    unsigned int cdata_size )
```

Decompress data via zlib.

Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>decompressed_size</code></td>
<td>exact size of the decompressed data to be returned</td>
</tr>
<tr>
<td><code>cdata</code></td>
<td>compressed data block</td>
</tr>
<tr>
<td><code>cdata_size</code></td>
<td>exact size of the compressed data block</td>
</tr>
</tbody>
</table>

Returns

Decompressed data block or NULL in low memory situations

Decompress the data from `cdata` of length `cdata_size` into a newly allocated block of size `decompressed_size` which is returned. The returned block needs to be released with `zintern_free()`. This function is intended to
decompress data which has been compressed with the packres.py utility, so no errors should occur during decompression. Consequently, if any error occurs during decompression or if the resulting data block is of a size other than decompressed size, the program will abort with an appropriate error message. If not enough memory could be allocated for decompression, NULL is returned.

```c
void Bse::zintern_free ( uint8 * dc_data )
```
Free data returned from `zintern_decompress()`.

### Variable Documentation

**cached_hash_secret**

```c
uint64_t Bse::cached_hash_secret
```
Use `hash_secret()` for access.

**KAMMER_FREQ**

```c
const SfiReal Bse::KAMMER_FREQ
```
Kammer note, representing the kammer frequency's MIDI note value for A' or A4.

**KAMMER_NOTE**

```c
const SfiInt Bse::KAMMER_NOTE
```
Value represents unparsable/unknown notes.

**KAMMER_OCTAVE**

```c
const SfiInt Bse::KAMMER_OCTAVE
```

**MAX_OCTAVE**

```c
const Bse::MAX_OCTAVE
```
Octave of MIN_NOTE.

**MIN_FINE_TUNE**

```c
const SfiInt Bse::MIN_FINE_TUNE
```
Octave of MAX_NOTE.

**MIN_OCTAVE**

```c
const Bse::MIN_OCTAVE
```
Octave number for MIDI A'.

### 1.2 Bse::AnsiColors Namespace Reference

The **AnsiColors** namespace contains utility functions for colored terminal output.
Enumerations

- enum Colors { , RESET }
  ANSI color symbols.

Functions

- void configure (Colorize colorize)
  Override the environment variable $BSE_COLOR (which may contain "always", "never" or "auto").
- bool colorize_tty (int fd)
  Check whether the tty fd should use colorization, checks BSE_COLOR if fd == -1.
- std::string color (Colors acolor, Colors c1, Colors c2, Colors c3, Colors c4, Colors c5, Colors c6)
  Return ANSI code for the specified color if stdout & stderr should be colorized, see colorize_tty().
- const char * color_code (Colors acolor)
  Return ANSI code for the specified color.

Detailed Description

The AnsiColors namespace contains utility functions for colored terminal output.

Enumeration Type Documentation

Colors

enum Bse::AnsiColors::Colors
ANSI color symbols.

Enumerator

| RESET | Reset combines BOLD_OFF, ITALICS_OFF, UNDERLINE_OFF, INVERSE_OFF, STRIKETHROUGH_OFF. |

Function Documentation

color()

std::string Bse::AnsiColors::color (Colors acolor, Colors c1, Colors c2, Colors c3, Colors c4, Colors c5, Colors c6)
Return ANSI code for the specified color if stdout & stderr should be colorized, see colorize_tty().

color_code()

const char * Bse::AnsiColors::color_code (Colors acolor)
Return ANSI code for the specified color.
colorize_tty()

```cpp
bool Bse::AnsiColors::colorize_tty (  
    int fd 
)  
```

Check whether the tty `fd` should use colorization, checks `BSE_COLOR` if `fd` == -1.

configure()

```cpp
void Bse::AnsiColors::configure (  
    Colorize colorize  
)  
```

Override the environment variable `$BSE_COLOR` (which may contain "always", "never" or "auto").

## 1.3 Bse::Lib Namespace Reference

Namespace for implementation internals.

### Classes

- **class KeccakF1600**
  
  The Keccak-φ[1600] Permutation, see the Keccak specification *Peeters und Assche (2011)*.

- **class ScopedLocale**
  
  Class to push a specific `locale_t` for the scope of its lifetime.

- **class ScopedPosixLocale**
  
  Class to push the POSIX/C `locale_t` (UTF-8) for the scope of its lifetime.

- **class StringFormatter**
  
  `StringFormatter` - `sprintf()` like string formatting for C++.

### Detailed Description

Namespace for implementation internals.

## 1.4 Bse::Path Namespace Reference

The **Path** namespace provides functions for file path manipulation and testing.

### Functions

- **String dirname** (const `String` &path)
  
  Retrieve the directory part of the filename @ path.

- **String basename** (const `String` &path)
  
  Strips all directory components from path and returns the resulting file name.

- **String realpath** (const `String` &path)
  
  Resolve links and directory references in path and provide a canonicalized absolute pathname.

- **String abspath** (const `String` &path, const `String` &incwd)

- **bool isabs** (const `String` &path)
  
  Return wether path is an absolute pathname.

- **bool isdirname** (const `String` &path)
  
  Return wether path is pointing to a directory component.

- **bool mkdirs** (const `String` &dirpath, `uint` mode)
  
  Create the directories in `dirpath` with `mode`, check `errno` on false returns.

- **String user_home** (const `String` &username)
  
  Get a user's home directory, uses `$HOME` if no username is given.

- **String data_home ()**
  
  Get the `XDG_DATA_HOME` directory, see: [https://specifications.freedesktop.org/basedir-spec/latest](https://specifications.freedesktop.org/basedir-spec/latest).
Detailed Description

The Path namespace provides functions for filepath manipulation and testing.

Function Documentation

abspath()

```cpp
String Bse::Path::abspath (  
    const String & path,  
    const String & incwd  
)
```

Parameters

<table>
<thead>
<tr>
<th>path</th>
<th>a filename path</th>
</tr>
</thead>
<tbody>
<tr>
<td>incwd</td>
<td>optional current working directory</td>
</tr>
</tbody>
</table>

Complete `path` to become an absolute file path. If necessary, `incwd` or the real current working directory is prepended.

basename()

```cpp
String Bse::Path::basename (  
    const String & path  
)
```
Strips all directory components from path and returns the resulting file name.

**cache_home()**

```cpp
String Bse::Path::cache_home();
```

Get the $XDG_CACHE_HOME directory, see: [https://specifications.freedesktop.org/basedir-spec/latest](https://specifications.freedesktop.org/basedir-spec/latest).

**check()**

```cpp
bool Bse::Path::check(const String &file, const String &mode);
```

**Parameters**

<table>
<thead>
<tr>
<th>file</th>
<th>possibly relative filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>mode</td>
<td>feature string</td>
</tr>
</tbody>
</table>

**Returns**

true if file adheres to mode

Perform various checks on file and return whether all checks passed. On failure, errno is set appropriately, and FALSE is returned. Available features to be checked for are:

- e/file must exist
- r/file must be readable
- w/file must be writable
- x/file must be executable
- f/file must be a regular file
- d/file must be a directory
- l/file must be a symbolic link
- c/file must be a character device
- b/file must be a block device
- p/file must be a named pipe
- s/file must be a socket.

**config_dirs()**

```cpp
String Bse::Path::config_dirs();
```

Get the $XDG_CONFIG_DIRS directory list, see: [https://specifications.freedesktop.org/basedir-spec/latest](https://specifications.freedesktop.org/basedir-spec/latest).

**config_home()**

```cpp
String Bse::Path::config_home();
```

Get the $XDG_CONFIG_HOME directory, see: [https://specifications.freedesktop.org/basedir-spec/latest](https://specifications.freedesktop.org/basedir-spec/latest).
config_names() [1/2]

String Bse::Path::config_names ()
Get config names as set with config_names(), if unset defaults to program_alias().

config_names() [2/2]

void Bse::Path::config_names (
    const String & names
)
Set a colon separated list of names for this application to find configuration settings and files.

cwd()

String Bse::Path::cwd ()
Return the current working directoy, including symlinks used in $PWD if available.

data_dirs()

String Bse::Path::data_dirs ()
Get the $XDG_DATA_DIRS directory list, see: https://specifications.freedesktop.org/basedir-spec/latest.

data_home()

String Bse::Path::data_home ()
Get the $XDG_DATA_HOME directory, see: https://specifications.freedesktop.org/basedir-spec/latest.

dirname()

String Bse::Path::dirname ( 
    const String & path
)
Retrieve the directory part of the filename @ path.

equals()

bool Bse::Path::equals ( 
    const String & file1, 
    const String & file2
)

Parameters

<table>
<thead>
<tr>
<th>file1</th>
<th>possibly relative filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>file2</td>
<td>possibly relative filename</td>
</tr>
</tbody>
</table>

Returns

TRUE if file1 and file2 are equal

Check whether file1 and file2 are pointing to the same inode in the same file system on the same device.

expand_tilde()

String Bse::Path::expand_tilde ( 
    const String & path
)
Expand a "~/" or "~user/" path which refers to user home directories.
1.4 Bse::Path Namespace Reference

isabs()

bool Bse::Path::isabs (const String & path)

Return whether path is an absolute pathname.

isdirname()

bool Bse::Path::isdirname (const String & path)

Return whether path is pointing to a directory component.

mkdirs()

bool Bse::Path::mkdirs (const String & dirpath, uint mode)

Create the directories in dirpath with mode, check errno on false returns.

realpath()

String Bse::Path::realpath (const String & path)

Resolve links and directory references in path and provide a canonicalized absolute pathname.

runtime_dir()

String Bse::Path::runtime_dir ()

Get the $XDG_RUNTIME_DIR directory, see: https://specifications.freedesktop.org/basedir-spec/latest.

searchpath_contains()

bool Bse::Path::searchpath_contains (const String & searchpath, const String & element)

Check if searchpath contains element, a trailing slash searches for directories.

searchpath_find()

String Bse::Path::searchpath_find (const String & searchpath, const String & file, const String & mode)

Find the first file in searchpath which matches mode (see check()).

searchpath_list()

StringVector Bse::Path::searchpath_list (const String & searchpath, const String & mode)

Find all searchpath entries matching mode (see check()).
searchpath_multiply()

```cpp
String Bse::Path::searchpath_multiply (  
    const String & searchpath,  
    const String & postfixes)
```

Yield a new searchpath by combining each element of `searchpath` with each element of `postfixes`.

user_home()

```cpp
String Bse::Path::user_home (  
    const String & username)
```

Get a user’s home directory, uses $HOME if no `username` is given.

1.5 Bse::Procedure Namespace Reference

The `Procedure` namespace contains procedure/IDL helpers.

Detailed Description

The `Procedure` namespace contains procedure/IDL helpers.

1.6 Bse::Re Namespace Reference

Some `std::regex` wrappers to simplify usage and reduce compilation time.

Detailed Description

Some `std::regex` wrappers to simplify usage and reduce compilation time.

1.7 Bse::Test Namespace Reference

The `Test` namespace offers utilities for unit tests.

Classes

- class `Timer`
  
  *Class for profiling benchmark tests.*

Functions

- void `init ( int *argcp, char **argv, const StringVector &args)`
  
  *Initialize the Bse core for a test program.*
- bool `slow ()`
  
  *Indicates whether slow tests should be run.*
- bool `verbose ()`
  
  *Indicates whether tests should run verbosely.*
- `uint64_t random_int64 ()`
  
  *Return random int for reproduceable tests.*
- `int64_t random_irange ( int64_t begin, int64_t end)`
  
  *Return random int within range for reproduceble tests.*
- `double random_float ()`
  
  *Return random double for reproduceble tests.*
- `double random_frange ( double begin, double end)`
  
  *Return random double within range for reproduceble tests.*
1.7 Bse::Test Namespace Reference

- **int run** (const *StringVector* &test_names)
  
  Run named tests.

- **int run ()**
  
  Run all registered tests.

- **String stringify_arg** (const *char* *a, const *char* *str_a)
  
  Stringify Args = =

### Detailed Description

The **Test** namespace offers utilities for unit tests.

The **Test** namespace is made available by `#include <bse/testing.hh>`

See also bse/testing.hh.

### Function Documentation

**init()**

```cpp
void Bse::Test::init (  
    int * argcp,  
    char ** argv,  
    const StringVector & args  
)
```

Initialize the Bse core for a test program.

Initializes Bse to execute unit tests by calling `parse_settings_and_args()` with args "autonomous = 1" and "testing = 1" and setting the application name. See also `#$BSE_DEBUG`.

**random_float()**

```cpp
double Bse::Test::random_float ()
```

Return random double for reproduceble tests.

**random_frange()**

```cpp
double Bse::Test::random_frange (  
    double begin,  
    double end  
)
```

Return random double within range for reproduceble tests.

**random_int64()**

```cpp
uint64_t Bse::Test::random_int64 ()
```

Return random int for reproduceble tests.

**random_irange()**

```cpp
int64_t Bse::Test::random_irange (  
    int64_t begin,  
    int64_t end  
)
```

Return random int within range for reproduceble tests.
run() [1/2]

```cpp
int Bse::Test::run (  
    const StringVector & test_names  
)
```

Run named tests.

run() [2/2]

```cpp
int Bse::Test::run (  
    void  
)
```

Run all registered tests.

slow()

```cpp
bool Bse::Test::slow (  
)
```

Indicates whether slow tests should be run.

stringify_arg()

```cpp
String Bse::Test::stringify_arg (  
    const char * a,  
    const char * str_a  
) [inline]  
== Stringify Args ==
```

verbose()

```cpp
bool Bse::Test::verbose (  
)
```

Indicates whether tests should run verbosely.

### 1.8 Bse::Xms Namespace Reference

Namespace for XML based serialization.

**Classes**

- **struct DataConverter**
  
  *Template to specialize XML attribute conversion for various data types.*

- **class Reflink**
  
  *Representation for an object reference between SerializableInterface objects.*

- **class SerializableInterface**
  
  *Interface for serializable objects with Reflink support.*

- **class SerializationField**
  
  *Reference to a storage attribute (or child node) for serialization.*

- **class SerializationNode**
  
  *Representation of a storage node for serialization via save() and load()*

**Detailed Description**

Namespace for XML based serialization.

### 1.9 Sfi Module Reference

The Sfi namespace contains utilities for synthesis.
Detailed Description

The `Sfi` namespace contains utilities for synthesis.
Chapter 2

Class Documentation

2.1 Bse::AlignedArray< T, ALIGNMENT > Class Template Reference

Class to maintain an array of aligned memory.
#include <bcore.hh>

Detailed Description

template<class T, int ALIGNMENT>
class Bse::AlignedArray< T, ALIGNMENT >

Class to maintain an array of aligned memory.
The documentation for this class was generated from the following file:
  • bse/bcore.hh

2.2 Bse::AlignedPOD< SIZE > Struct Template Reference

Helper to provide memory for placement new AlignedPOD<SIZE> is aligned like max_align_t or like malloc()-ed memory and provides SIZE bytes.
#include <randomhash.hh>

Detailed Description

template<size_t SIZE>
struct Bse::AlignedPOD< SIZE >

Helper to provide memory for placement new AlignedPOD<SIZE> is aligned like max_align_t or like malloc()-ed memory and provides SIZE bytes.
Idiomatic use is:

static AlignedPOD<sizeof (std::string)> pod_mem;
std::string *str = new (&pod_mem) std::string();

The documentation for this struct was generated from the following file:
  • bse/randomhash.hh

2.3 Bse::AsyncBlockingQueue< Value > Class Template Reference

Asynchronous queue to push/pop values across thread boundaries.
#include <bcore.hh>
### Detailed Description

```cpp
template<class Value>
class Bse::AsyncBlockingQueue<Value>
```

Asyncronous queue to push/pop values across thread boundaries. The `AsyncBlockingQueue` is a thread-safe asynchronous queue which blocks in `pop()` until data is provided through `push()` from any thread.

The documentation for this class was generated from the following file:

- bse/bcore.hh

### 2.4 Bse::AutoSeeder Class Reference

`AutoSeeder` provides non-deterministic seeding entropy.

```cpp
#include <randomhash.hh>
```

### Public Member Functions

- `uint64 operator() () const`  
  Generate non-deterministic 64bit random value.

- `template<typename RandomAccessIterator > void generate(RandomAccessIterator begin, RandomAccessIterator end)`  
  Fill the range `begin, end) with random unsigned integer values.

### Static Public Member Functions

- `static uint64 random ()`  
  Generate non-deterministic 64bit random value.

### Detailed Description

`AutoSeeder` provides non-deterministic seeding entropy.

### Member Function Documentation

#### generate()

```cpp
template<typename RandomAccessIterator >
void Bse::AutoSeeder::generate(
    RandomAccessIterator begin,
    RandomAccessIterator end ) [inline]
```

Fill the range `begin, end) with random unsigned integer values.

#### operator()

```cpp
uint64 Bse::AutoSeeder::operator() () const [inline]
```

Generate non-deterministic 64bit random value.

#### random()

```cpp
static uint64 Bse::AutoSeeder::random() [inline], [static]
```

Generate non-deterministic 64bit random value.

The documentation for this class was generated from the following file:

- bse/randomhash.hh
2.5 Bse::AuxData Struct Reference

**AuxData** - record to describe entity attributes with "key = value" strings.
import"bseapi.idl";
Inherited by Bse::AuxDataAndIcon.

**Public Attributes**

- **String entity**
  
  *Entity that has an auxiliary data list.*

- **StringSeq attributes**
  
  *List of "key = value" auxiliary data strings.*

**Detailed Description**

**AuxData** - record to describe entity attributes with "key = value" strings.

**Member Data Documentation**

**attributes**

**StringSeq Bse::AuxData::attributes**
List of "key = value" auxiliary data strings.

**entity**

**String Bse::AuxData::entity**
Entity that has an auxiliary data list.
The documentation for this struct was generated from the following file:

- bse/bseapi.idl

2.6 Bse::AuxDataSeq Struct Reference

**AuxDataSeq** - a variable length list of **AuxData** records.
import"bseapi.idl";

**Detailed Description**

**AuxDataSeq** - a variable length list of **AuxData** records.
The documentation for this struct was generated from the following file:

- bse/bseapi.idl

2.7 Bse::Blob Class Reference

Binary large object storage container.
#include <blob.hh>
Public Member Functions

- **String name ()**
  
  Retrieve the Blob's filename or url.

- **const char * data ()**
  
  Retrieve the Blob's data.

- **const uint8 * bytes ()**
  
  Retrieve the Blob's data as uint8 buffer.

- **size_t size ()**
  
  Retrieve the Blob's data size in bytes.

- **String string ()**
  
  Copy Blob data into a zero terminated string.

- **Blob ()**
  
  Construct an empty Blob.

- **Blob (const String &auto_url)**
  
  Construct Blob from url or filename (auto detected).

- **operator bool ()** const
  
  Checks if the Blob contains accessible data.

Static Public Member Functions

- **static Blob from_file (const String &filename)**
  
  Create Blob by loading from filename.

- **static Blob from_url (const String &url)**
  
  Create Blob by opening a url.

Detailed Description

Binary large object storage container.

Constructor & Destructor Documentation

**Blob() [1/2]**

Bse::Blob::Blob ( ) [explicit]

Construct an empty Blob.

**Blob() [2/2]**

Bse::Blob::Blob ( const String & auto_url ) [explicit]

Construct Blob from url or filename (auto detected).

Member Function Documentation

**bytes()**

const uint8 * Bse::Blob::bytes ( )

Retrieve the Blob's data as uint8 buffer.
data()

const char * Bse::Blob::data () const
Retrieve the Blob's data.

from_file()

Blob Bse::Blob::from_file (const String & filename ) [static]
Create Blob by loading from filename.

from_url()

Blob Bse::Blob::from_url (const String & url ) [static]
Create Blob by opening a url.

name()

String Bse::Blob::name ()
Retrieve the Blob's filename or url.

operator bool()

Bse::Blob::operator bool () const [explicit]
Checks if the Blob contains accessible data.

size()

size_t Bse::Blob::size ()
Retrieve the Blob's data size in bytes.

string()

String Bse::Blob::string ()
Copy Blob data into a zero terminated string.
The documentation for this class was generated from the following files:

- bse/blob.hh
- bse/blob.cc

2.8 Bse::Bus Interface Reference

Interface for effect stacks and per-track audio signal routing to the master output.
import"bseapi.idl";
Inheritance diagram for Bse::Bus:

![Inheritance Diagram]

### Public Member Functions

- **Error** `ensure_output ()`
  
  *Ensure that a bus has an output connection.*

- **Error** `connect_bus (Bus bus)`
  
  *Add a bus to the input list of a bus.*

- **Error** `connect_track (Track track)`
  
  *Add a track to the input list of a bus.*

- **Error** `disconnect_bus (Bus bus)`
  
  *Remove a bus from the input list of a bus.*
Remove a bus from the input list of a bus.

- Error `disconnect_track (Track track)`
  Remove a track from the input list of a bus.

### Detailed Description

Interface for effect stacks and per-track audio signal routing to the master output.

### Member Function Documentation

**connect_bus()**

```cpp
Error Bse::Bus::connect_bus (Bus bus)
```
Add a bus to the input list of a bus.

**connect_track()**

```cpp
Error Bse::Bus::connect_track (Track track)
```
Add a track to the input list of a bus.

**disconnect_bus()**

```cpp
Error Bse::Bus::disconnect_bus (Bus bus)
```
Remove a bus from the input list of a bus.

**disconnect_track()**

```cpp
Error Bse::Bus::disconnect_track (Track track)
```
Remove a track from the input list of a bus.

**ensure_output()**

```cpp
Error Bse::Bus::ensure_output ()
```
Ensure that a bus has an output connection.

The documentation for this interface was generated from the following file:

- bse/bseapi.idl

### 2.9 Bse::BusIface Class Reference

IDL interface class for **Bse::Bus**.
#include <bseapi_interfaces.hh>
Inheritance diagram for Bse::BusIface:

![Inheritance Diagram](image)

**Additional Inherited Members**

**Detailed Description**

IDL interface class for Bse::Bus.

The documentation for this class was generated from the following file:

- bse/bseapi_interfaces.hh
### 2.10 Bse::BusImpl Class Reference

Inheritance diagram for Bse::BusImpl:

![Inheritance Diagram](image)

**Additional Inherited Members**

**Detailed Description**

The documentation for this class was generated from the following files:

- bse/bsebus.hh
- bse/bsebus.cc

---

### 2.11 Bse::Category Struct Reference

Categories describe useful type entities.
2.12 Bse::CategorySeq Struct Reference

Sequence of Category records.

import "bseapi.idl";

Detailed Description
Categories describe useful type entities.
The documentation for this struct was generated from the following file:

- bse/bseapi.idl

2.13 Bse::Container Interface Reference

Base interface type for containers of Item derived types.

import "bseapi.idl";

Inheritance diagram for Bse::Container:

[Diagram showing the inheritance hierarchy of Bse::Container and its subtypes with various methods and attributes labeled.]
Public Member Functions

- **Item lookup_item (String uname)**
  
  Find an immediate child of a container by name (unique per container child).

- **Item get_item (String item_type, int32 seq_id)**
  
  Retrieve the immediate child of type item_type by its sequential id (the 'nth' child).

- **ItemSeq list_children ()**
  
  List all immediate children of a container.

Detailed Description

Base interface type for containers of Item derived types.

Events:

- **treechange** - Detail: 'additem' or 'removeitem' - a notification event is sent when an item is added to or removed from the container.

Member Function Documentation

**get_item()**

Item Bse::Container::get_item (  
    String item_type,  
    int32 seq_id  
)

Retrieve the immediate child of type item_type by its sequential id (the 'nth' child).

**list_children()**

ItemSeq Bse::Container::list_children ()

List all immediate children of a container.

**lookup_item()**

Item Bse::Container::lookup_item (  
    String uname  
)

Find an immediate child of a container by name (unique per container child).

The documentation for this interface was generated from the following file:

- bse/bseapi.idl

2.14 Bse::Container_iface Class Reference

IDL interface class for Bse::Container.

#include <bseapi_interfaces.hh>
Inheritance diagram for Bse::ContainerIface:

```
+ ContainerImpl()
+ lookup_item()
+ get_item()
+ list_children()
```

**Additional Inherited Members**

**Detailed Description**

IDL interface class for `Bse::Container`.
The documentation for this class was generated from the following file:

- `bse/bseapi_interfaces.hh`
2.15  Bse::ContainerImpl Class Reference

Inheritance diagram for Bse::ContainerImpl:

![Inheritance Diagram]

Additional Inherited Members

Detailed Description

The documentation for this class was generated from the following files:

- bse/bsecontainer.hh
- bse/bsecontainer.cc

2.16  Bse::ContextMerger Interface Reference

Source module for merging multiple synthesis contexts, used to implement polyphony.

import "bseapi.idl";
Inheritance diagram for Bse::ContextMerger:

```
Bse::Container
+ lookup_item()
+ get_item()
+ list_children()

Bse::ContextMerger

Bse::ContextMergerIface

Bse::ContextMergerImpl
+ ContextMergerImpl()

Bse::Track
+ outputs
+ get_timing()
+ insert_part()
+ remove_tick()
+ remove_link()
+ list_devices()
+ create_device()
+ device_type_info()
+ list_device_types()
+ list_parts_uniq()
+ list_parts()
and 7 more...

Bse::ContextMerger
Bse::ContextMergerIface
Bse::ContextMergerImpl
+ ContextMergerImpl()

Bse::Track
+ outputs
+ ... list_parts_uniq()
+ list_parts()
and 7 more...

Bse::Container
+ lookup_item()
+ get_item()
+ list_children()
```

Additional Inherited Members

Detailed Description

Source module for merging multiple synthesis contexts, used to implement polyphony. The documentation for this interface was generated from the following file:

- bse/bseapi.idl

2.17 Bse::ContextMergerIface Class Reference

IDL interface class for Bse::ContextMerger.
#include <bseapi_interfaces.hh>

Inheritance diagram for Bse::ContextMergerIface:

```
Bse::ContextMergerIface
    Bse::ContextMerger
    Bse::ContextMergerImpl

+ ContextMergerImpl()
```

**Additional Inherited Members**

**Detailed Description**

IDL interface class for Bse::ContextMerger.
The documentation for this class was generated from the following file:

- bse/bseapi_interfaces.hh
2.18 Bse::ContextMergerImpl Class Reference

Inheritance diagram for Bse::ContextMergerImpl:

Additional Inherited Members

Detailed Description

The documentation for this class was generated from the following files:

- bse/bsecontextmerger.hh
- bse/bsecontextmerger.cc
2.19 ConvertAny Struct Reference

Convert between Aida::Any and Jsonipc::JsonValue.

```cpp
#include <beast-sound-engine.hh>
```

Inherited by Jsonipc::Convert< Aida::Any >.

**Detailed Description**

Convert between Aida::Any and Jsonipc::JsonValue.

The documentation for this struct was generated from the following file:
- bse/beast-sound-engine.hh

2.20 Bse::CSynth Interface Reference

Customizable synthesis (filter) network container.

```cpp
import "bseapi.idl";
```

Inheritance diagram for Bse::CSynth:
Additional Inherited Members

Detailed Description

Customizable synthesis (filter) network container.
The documentation for this interface was generated from the following file:

- bse/bseapi.idl

2.21  Bse::CSynthIface Class Reference

IDL interface class for Bse::CSynth.

```cpp
#include <bseapi_interfaces.hh>
```

Inheritance diagram for Bse::CSynthIface:

```
Bse::CSynth
```
```
Bse::CSynthIface
```
```
Bse::CSynthImpl
```
```
+ CSynthImpl()
```

Additional Inherited Members

Detailed Description

IDL interface class for Bse::CSynth.
The documentation for this class was generated from the following file:

- bse/bseapi_interfaces.hh
2.22 Bse::CSynthImpl Class Reference

Inheritance diagram for Bse::CSynthImpl:

Additional Inherited Members

Detailed Description

The documentation for this class was generated from the following files:

- bse/bsecsynth.hh
- bse/bsecsynth.cc

2.23 Bse::Xms::DataConverter< T, typename > Struct Template Reference

Template to specialize XML attribute conversion for various data types.
#include <serializable.hh>

Detailed Description

template<typename T, typename = void>
struct Bse::Xms::DataConverter< T, typename >

Template to specialize XML attribute conversion for various data types.
The documentation for this struct was generated from the following file:

- bse/serializable.hh

## 2.24 Bse::DataKey< Type > Class Template Reference

**DataKey** objects are used to identify and manage custom data members of **DataListContainer** objects.

### #include <datalist.hh>

### Public Member Functions

- **virtual** Type **fallback()**
  
  *Return the default value for Type.*

- **virtual** void **destroy**(Type data)
  
  *Hook invoked when data is deleted.*

### Detailed Description

**template<typename Type>**

**class Bse::DataKey< Type >**

**DataKey** objects are used to identify and manage custom data members of **DataListContainer** objects.

### Member Function Documentation

#### destroy()

**template<typename Type>**

**virtual** void **Bse::DataKey< Type >::destroy**(Type data) [inline], [virtual]

Hook invoked when **data** is deleted.

#### fallback()

**template<typename Type>**

**virtual** Type **Bse::DataKey< Type >::fallback**( ) [inline], [virtual]

*Return the default value for Type.*

The documentation for this class was generated from the following file:

- bse/datalist.hh

## 2.25 Bse::DataList Class Reference

Underlying storage implementation for a **DataListContainer**.

### #include <datalist.hh>

### Detailed Description

Underlying storage implementation for a **DataListContainer**.

The documentation for this class was generated from the following files:

- bse/datalist.hh
- bse/datalist.cc
2.26 Bse::DataListContainer Class Reference

DataListContainer - typesafe storage and retrieval of arbitrary members.

```
#include <datalist.hh>
```

Inheritance diagram for Bse::DataListContainer:

```
Bse::DataListContainer
+ set_data()
+ get_data()
+ swap_data()
+ delete_data()
* set_data()
* get_data()
* swap_data()
* swap_data()
* delete_data()

Bse::LegacyObjectImpl
+ LegacyObjectImpl()
+ ~LegacyObjectImpl()
+ operator BseObject *()
+ debug_name()
+ unique_id()
+ uname()
+ uname()
+ proxy_id()
+ find_typedata()
+ as_bse_object()
```

Public Member Functions

Accessing custom data members

- template<typename Type>
  void set_data (DataKey < Type > *key, Type data)

  *Assign data to the custom keyed data member, deletes any previously set data.*

- template<typename Type>
  Type get_data (DataKey < Type > *key) const

  *Retrieve contents of the custom keyed data member, returns DataKey::fallback if nothing was set.*

- template<typename Type>
  Type swap_data (DataKey < Type > *key, Type data)

  *Swap data with the current contents of the custom keyed data member, returns the current contents.*

- template<typename Type>
  Type swap_data (DataKey < Type > *key)

  *Removes and returns the current contents of the custom keyed data member without deleting it.*
Detailed Description

DataListContainer - typesafe storage and retrieval of arbitrary members. By using a DataKey, DataListContainer objects allow storage and retrieval of custom data members in a typesafe fashion. The custom data members will initially default to DataKey::fallback and are deleted by the DataListContainer destructor. Example:

Member Function Documentation

delete_data()

```cpp
template<typename Type >
void Bse::DataListContainer::delete_data ( 
    DataKey< Type > * key ) const [inline]
```
Delete the current contents of the custom keyed data member, invokes DataKey::destroy.

get_data()

```cpp
template<typename Type >
Type Bse::DataListContainer::get_data ( 
    DataKey< Type > * key ) const [inline]
```
Retrieve contents of the custom keyed data member, returns DataKey::fallback if nothing was set.

set_data()

```cpp
template<typename Type >
void Bse::DataListContainer::set_data ( 
    DataKey< Type > * key, 
    Type data ) [inline]
```
Assign data to the custom keyed data member, deletes any previously set data.

swap_data() [1/2]

```cpp
template<typename Type >
Type Bse::DataListContainer::swap_data ( 
    DataKey< Type > * key, 
    Type data ) [inline]
```
Swap data with the current contents of the custom keyed data member, returns the current contents.

swap_data() [2/2]

```cpp
template<typename Type >
Type Bse::DataListContainer::swap_data ( 
    DataKey< Type > * key ) [inline]
```
Removes and returns the current contents of the custom keyed data member without deleting it. The documentation for this class was generated from the following file:

- bse/datalist.hh
2.27   **Bse::Device Interface Reference**

Interface for the encapsulation of audio processors.

```cpp
import "bseapi.idl";
```

Inheritance diagram for Bse::Device:

```
<table>
<thead>
<tr>
<th>Bse::Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ set_prop()</td>
</tr>
<tr>
<td>+ get_prop()</td>
</tr>
<tr>
<td>+ find_prop()</td>
</tr>
<tr>
<td>+ list_props()</td>
</tr>
<tr>
<td>+ notify()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bse::Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ list_modules()</td>
</tr>
<tr>
<td>+ create_module()</td>
</tr>
<tr>
<td>+ module_type_info()</td>
</tr>
<tr>
<td>+ list_module_types()</td>
</tr>
<tr>
<td>+ get_device_type()</td>
</tr>
<tr>
<td>+ device_type_info()</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Bse::DeviceIface</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Bse::DeviceImpl</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ DeviceImpl()</td>
</tr>
<tr>
<td>+ list_modules()</td>
</tr>
<tr>
<td>+ create_module()</td>
</tr>
<tr>
<td>+ module_type_info()</td>
</tr>
<tr>
<td>+ list_module_types()</td>
</tr>
<tr>
<td>+ get_device_type()</td>
</tr>
<tr>
<td>+ device_type_info()</td>
</tr>
<tr>
<td>+ device_type_info()</td>
</tr>
<tr>
<td>+ list_device_types()</td>
</tr>
</tbody>
</table>
```

**Public Member Functions**

- **ModuleSeq list_modules ()**
  
  *List modules in order of processing.*

- **Module create_module (String module_id)**
Create a new module with module_type.

- **ModuleTypeInfo module_type_info (String module_id)**
  
  Describe module_type.

- **StringSeq list_module_types ()**
  
  List known module types.

- **String get_device_type ()**
  
  Retrieve type of device to be created.

- **DeviceTypeInfo device_type_info ()**
  
  Describe this device type.

**Detailed Description**

Interface for the encapsulation of audio processors.

**Member Function Documentation**

**create_module()**

_module Bse::Device::create_module (String module_id)_

Create a new module with module_type.

**device_type_info()**

_deviceTypeInfo Bse::Device::device_type_info ()_

Describe this device type.

**get_device_type()**

_string Bse::Device::get_device_type ()_

Retrieve type of device to be created.

**list_module_types()**

_stringSeq Bse::Device::list_module_types ()_

List known module types.

**list_modules()**

_moduleSeq Bse::Device::list_modules ()_

List modules in order of processing.

**module_type_info()**

_moduleTypeInfo Bse::Device::module_type_info (String module_id)_

Describe module_type.

The documentation for this interface was generated from the following file:

- bse/bseapi.idl
2.28 Bse::DeviceCrawlerIface Class Reference

IDL interface class for Bse::DeviceCrawler.
#include <bseapi_interfaces.hh>
Inheritance diagram for Bse::DeviceCrawlerIface:

![Inheritance Diagram](image)

### Detailed Description

IDL interface class for Bse::DeviceCrawler.
The documentation for this class was generated from the following file:

- `bse/bseapi_interfaces.hh`
2.29 Bse::DeviceCrawlerImpl Class Reference

Inheritance diagram for Bse::DeviceCrawlerImpl:

![Inheritance Diagram]

### Additional Inherited Members

### Detailed Description

The documentation for this class was generated from the following files:

- bse/devicecrawler.hh
- bse/devicecrawler.cc

2.30 Bse::DeviceIface Class Reference

IDL interface class for Bse::Device.

#include <bseapi_interfaces.hh>
Inheritance diagram for Bse::DeviceIntace:

```
Bse::Device

+ list_modules()
+ create_module()
+ module_type_info()
+ list_module_types()
+ get_device_type()
+ device_type_info()

Bse::DeviceImpl

+ DeviceImpl()
+ list_modules()
+ create_module()
+ module_type_info()
+ list_module_types()
+ get_device_type()
+ device_type_info()
+ list_device_types()
```

Additional Inherited Members

Detailed Description

IDL interface class for Bse::Device.
The documentation for this class was generated from the following file:

- bse/bseapi_interfaces.hh
2.31 Bse::DeviceImpl Class Reference

Inheritance diagram for Bse::DeviceImpl:

Additional Inherited Members

Detailed Description

The documentation for this class was generated from the following files:

- bse/device.hh
- bse/device.cc
2.32  Bse::DeviceTypeInfo Struct Reference

Info for device types.
import"bseapi.idl";

Detailed Description
Info for device types.
The documentation for this struct was generated from the following file:

- bse/bseapi.idl

2.33  Bse::DriverEntry Struct Reference

Driver information for PCM and MIDI handling.
import"bseapi.idl";

Detailed Description
Driver information for PCM and MIDI handling.
The documentation for this struct was generated from the following file:

- bse/bseapi.idl

2.34  Bse::DriverEntrySeq Struct Reference

DriverEntry sequence.
import"bseapi.idl";

Detailed Description
DriverEntry sequence.
The documentation for this struct was generated from the following file:

- bse/bseapi.idl

2.35  Bse::EditableSample Interface Reference

Interface for editable PCM wave samples.
import"bseapi.idl";
Inheritance diagram for Bse::EditableSample:

![Inheritance diagram](image)

**Public Member Functions**

- **FloatSeq collect_stats** (int64 `voffset`, float64 `offset_scale`, int64 `block_size`, int64 `stepping`, int64 `max_pairs`)
  
  *Collect statistics from sample blocks as (minimum, maximum) pairs.*

- **void close** ()
  
  *Close an opened sample.*

- **int64 get_length** ()
  
  *Return the number of values in the sample.*
• **int64 get_n_channels ()**
  
  Return the number of channels in the sample.

• **float64 get_osc_freq ()**
  
  Return the oscillator frequency for the sample.

• **Error open ()**
  
  Open the sample for reading.

### Detailed Description

Interface for editable PCM wave samples.

### Member Function Documentation

**close()**

```cpp
void Bse::EditableSample::close ( )
```

Close an opened sample.

**collect_stats()**

```cpp
FloatSeq Bse::EditableSample::collect_stats ( 
    int64 voffset,
    float64 offset_scale,
    int64 block_size,
    int64 stepping,
    int64 max_pairs )
```

Collect statistics from sample blocks as (minimum, maximum) pairs.

**Parameters**

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>voffset</strong></td>
<td>Offset of first stat block</td>
</tr>
<tr>
<td><strong>offset-scale</strong></td>
<td>Factor to scale voffset increments with</td>
</tr>
<tr>
<td><strong>block-size</strong></td>
<td>Block size to compute stat pairs from</td>
</tr>
<tr>
<td><strong>stepping</strong></td>
<td>Stepping within a stat block</td>
</tr>
<tr>
<td><strong>max-pairs</strong></td>
<td>Maximum number of (min, max) pairs to collect</td>
</tr>
</tbody>
</table>

**Returns**

Block of samples

**get_length()**

```cpp
int64 Bse::EditableSample::get_length ( )
```

Return the number of values in the sample.

**get_n_channels()**

```cpp
int64 Bse::EditableSample::get_n_channels ( )
```

Return the number of channels in the sample.
get_osc_freq()
float64 Bse::EditableSample::get_osc_freq ( )
Return the oscillator frequency for the sample.

open()
Error Bse::EditableSample::open ( )
Open the sample for reading.
The documentation for this interface was generated from the following file:
  • bse/bseapi.idl

2.36  Bse::EditableSampleIface Class Reference

IDL interface class for Bse::EditableSample.
#include <bseapi_interfaces.hh>
Inheritance diagram for Bse::EditableSampleIface:
Additional Inherited Members

Detailed Description

IDL interface class for Bse::EditableSample.
The documentation for this class was generated from the following file:

- bse/bseapi_interfaces.hh

### 2.37 Bse::EditableSampleImpl Class Reference

Inheritance diagram for Bse::EditableSampleImpl:
Additional Inherited Members

Detailed Description

The documentation for this class was generated from the following files:

- bse/bseeditablesample.hh
- bse/bseeditablesample.cc

2.38 Bse::Flac1Handle Class Reference

Flac1Handle supports storing flac files as binary appendix to BSE files.

```cpp
#include <bsedatahandle-flac.hh>
```

Public Member Functions

- `int read_data(void *buffer, uint blength)`
  Returns `-errno || length`.
- `void put_wstore(SfiWStore *wstore)`
  This function deletes the `flac1handle` object when `sfi_wstore_destroy(wstore)` is executed.

Static Public Member Functions

- `static Flac1Handle * create(GslDataHandle *dhandle)`
  Return a valid `Flac1Handle` if `dhandle` is not flac, and a `Flac1Handle` otherwise.

Detailed Description

Flac1Handle supports storing flac files as binary appendix to BSE files.

Member Function Documentation

**create()**

```cpp
Flac1Handle * Flac1Handle::create ( 
    GslDataHandle * dhandle ) [static]
```

Return a valid `Flac1Handle` if `dhandle` is not flac, and a `Flac1Handle` otherwise.

**put_wstore()**

```cpp
void Flac1Handle::put_wstore ( 
    SfiWStore * wstore )
```

This function deletes the `flac1handle` object when `sfi_wstore_destroy(wstore)` is executed.

**read_data()**

```cpp
int Flac1Handle::read_data ( 
    void * buffer, 
    uint blength )
```

Returns `-errno || length`.

The documentation for this class was generated from the following files:

- bse/bsedatahandle-flac.hh
- bse/bsedatahandle-flac.cc
2.39  Bse::FloatSeq Struct Reference

A list of floating point values.

import "bseapi.idl";

Detailed Description

A list of floating point values.
The documentation for this struct was generated from the following file:

• bse/bseapi.idl

2.40  Bse::FriendAllocator< T > Class Template Reference

A std::make_shared<>() wrapper class to access private ctor & dtor.

#include <cxxaux.hh>

Inheritance diagram for Bse::FriendAllocator< T >:

```
std::allocator< T >
+ elements
+ address()
+ allocate()
+ allocator()
+ construct()
+ deallocate()
+ destroy()
+ max_size()
+ ~allocator()

Bse::FriendAllocator< T >
+ construct()
+ destroy()
+ make_shared()
```

Static Public Member Functions

• template<typename C , typename... Args>
  static void construct (C *p, Args &... args)
  
  *Construct type C object, standard allocator requirement.*

• template<typename C >
  static void destroy (C *p)
  
  *Delete type C object, standard allocator requirement.*

• template<typename ... Args>
  static std::shared_ptr< T > make_shared (Args &... args)
  
  *Construct an object of type T that is wrapped into a std::shared_ptr<T>.*
Additional Inherited Members

Detailed Description

template<class T>
class Bse::FriendAllocator<T>

A `std::make_shared<>()` wrapper class to access private ctor & dtor.
To call `std::make_shared<T>()` on a class `T`, its constructor and destructor must be public. For classes with
private or protected constructor or destructor, this class can be used as follows:

```cpp
class Type {
    Type (ctor_args...); // Private ctor.
    friend class FriendAllocator<Type>; // Allow access to ctor/dtor of Type.
};
std::shared_ptr<Type> t = FriendAllocator<Type>::make_shared (ctor_args...);
```

Member Function Documentation

`construct()`

```cpp
template<class T>
template<typename C , typename... Args>
static void Bse::FriendAllocator<T>::construct ( C * p,
    Args &&... args ) [inline], [static]
```
Construct type `C` object, standard allocator requirement.

`destroy()`

```cpp
template<class T>
template<typename C>
static void Bse::FriendAllocator<T>::destroy ( C * p ) [inline], [static]
```
Delete type `C` object, standard allocator requirement.

`make_shared()`

```cpp
template<class T>
template<typename ... Args>
static std::shared_ptr<T> Bse::FriendAllocator<T>::make_shared ( Args &&... args ) [inline], [static]
```
Construct an object of type `T` that is wrapped into a `std::shared_ptr<T>`.

Parameters

| `args` | The list of arguments to pass into a T() constructor. |

Returns

A `std::shared_ptr<T>` owning the newly created object.

The documentation for this class was generated from the following file:

- bse/cxxaux.hh

2.41 Bse::Icon Struct Reference

Representation of an icon pixel image.
2.42 Bse::Item Interface Reference

Base interface type for objects that can be added to a container.

Public Member Functions

- **Item use ()**
  
  Increment use count to keep an item alive.

- **void unuse ()**
  
  Decrement use count for when an item is not needed anymore.

- **void set_name (String name)**
  
  Assign an item’s name.

- **Item common_ancestor (Item other)**
  
  Find a common container (parent or grand-parent) of two items if any.

- **bool check_is_a (String type_name)**
  
  Check whether an item has a certain type.
• void **group**_undo**( String group_name)**
  Request multiple modifying actions on an item to be grouped together as a single undo operation.

• void **ungroup undo**()
  Ends the undo grouping opened up by a previous group-undo() call.

• **Project get_project**()
  Retrieve an item's project.

• **Item get_parent**()
  Retrieve an item's parent.

• int32 **get_seqid**()
  Retrieve an item's sequential ID. The sequential ID depends on the item's type and its position in between siblings of the same type within its immediate container.

• **String get_type**()
  Retrieve an item's type name.

• **String get_type_authors**()
  Retrieve authors of an item's type implementation.

• **String get_type_blurb**()
  Retrieve an item's type description.

• **String get_type_license**()
  Retrieve the license for an item's type implementation.

• **String get_type_name**()
  Retrieve an item's type name.

• **String get_uname_path**()
  Retrieve the project relative uname path for this item.

• **String get_name**()
  Retrieve an item's name.

• **String get_name_or_type**()
  Retrieve an item's name or type if it has no name.

• bool **internal**()
  Check whether an item is internal, i.e. owned by another non-internal item.

• bool **editable_property**( String property)
  Test whether a property is editable according to object state and property options.

• PropertyCandidates **get_property_candidates**( String property_name)
  Retrieve tentative values for an item or item sequence property.

### Detailed Description
Base interface type for objects that can be added to a container.

### Member Function Documentation

**check_is_a()**

```cpp
bool Bse::Item::check_is_a ( String type_name )
```
Check whether an item has a certain type.

**common_ancestor()**

```cpp
Item Bse::Item::common_ancestor ( Item other )
```
Find a common container (parent or grand-parent) of two items if any.
editable_property()

bool Bse::Item::editable_property (  
    String property )
Test whether a property is editable according to object state and property options.

get_name()

String Bse::Item::get_name ( )
Retrieve an item's name.

get_name_or_type()

String Bse::Item::get_name_or_type ( )
Retrieve an item's name or type if it has no name.

get_parent()

Item Bse::Item::get_parent ( )
Retrieve an item's parent.

get_project()

Project Bse::Item::get_project ( )
Retrieve an item's project.

get_property_candidates()

PropertyCandidates Bse::Item::get_property_candidates (  
    String property_name )
Retrieve tentative values for an item or item sequence property.

get_seqid()

int32 Bse::Item::get_seqid ( )
Retrieve an item's sequential ID. The sequential ID depends on the item's type and its position inbetween siblings of the same type within its immediate container.

get_type()

String Bse::Item::get_type ( )
Retrieve an item's type name.

get_type_authors()

String Bse::Item::get_type_authors ( )
Retrieve authors of an item's type implementation.
2.42 Bse::Item Interface Reference

get_type_blurb()

String Bse::Item::get_type_blurb ( )
Retrieve an item's type description.

get_type_license()

String Bse::Item::get_type_license ( )
Retrieve the license for an item's type implementation.

get_type_name()

String Bse::Item::get_type_name ( )
Retrieve an item's type name.

group_undo()

void Bse::Item::group_undo ( 
    String group_name )
Request multiple modifying actions on an item to be grouped together as a single undo operation.

internal()

bool Bse::Item::internal ( )
Check whether an item is internal, i.e. owned by another non-internal item.

set_name()

void Bse::Item::set_name ( 
    String name )
Assign an item's name.

ungroup_undo()

void Bse::Item::ungroup_undo ( )
Ends the undo grouping opened up by a previous group-undo() call.

unuse()

void Bse::Item::unuse ( )
Decrement use count for when an item is not needed anymore.
use()

Item Bse::Item::use()
Increment use count to keep an item alive.
The documentation for this interface was generated from the following file:
  • bse/bseapi.idl

2.43 Bse::ItemIface Class Reference

IDL interface class for Bse::Item.
#include <bseapi_interfaces.hh>
Inheritance diagram for Bse::ItemIface:

```
Bse::Item
+ icon
+ seqid
+ use()
+ unuse()
+ set_name()
+ common_ancestor()
+ check_is_a()
+ group_undo()
+ ungroup_undo()
+ get_project()
+ get_parent()
+ get_seqid()
and 11 more...

Bse::ItemIface

Bse::ItemImpl
+ ItemImpl()
+ parent()
+ use()
+ unuse()
+ set_name()
+ editable_property()
+ icon()
+ icon()
+ common_ancestor()
+ check_is_a()
and 25 more...
```

**Additional Inherited Members**

**Detailed Description**

IDL interface class for Bse::Item.
The documentation for this class was generated from the following file:

- bse/bseapi_interfaces.hh
2.44 Bse::ItemImpl Class Reference

Inheritance diagram for Bse::ItemImpl:

Classes

- class UndoDescriptor

  UndoDescriptor - type safe object handle to persist undo/redo steps.

Public Member Functions

- void push_property_undo (const String &property_name)
  
  Save the value of property name onto the undo stack.

- template<typename ItemT, typename... FuncArgs, typename... CallArgs>
  void push_undo (const String &blurb, ItemT &self, Error(ItemT::*function)(FuncArgs...), CallArgs... args)

  Push an undo function onto the undo stack, the self argument to function must match this.

- template<typename ItemT, typename R, typename... FuncArgs, typename... CallArgs>
  void push_undo (const String &blurb, ItemT &self, R(ItemT::*function)(FuncArgs...), CallArgs... args)

  Push an undo function like push_undo(), but ignore the return value of function.

- template<typename ItemT, typename ItemTLambda>
  void push_undo (const String &blurb, ItemT &self, const ItemTLambda &itemt_lambda)

  Push an undo lambda, using the signature: Error lambda (TypeDerivedFromItem&, BseUndoStack*);
• template<typename ItemT, typename ItemTLambda>
  void push_undo_to_redo(const String &blurb, ItemT &self, const ItemTLambda &itemt_lambda)
  Push an undo step, that when executed, pushes itemt_lambda to the redo stack.

• template<class Obj>
  UndoDescriptor<Obj> undo_descriptor(Obj &item)
  Create an object descriptor that persists undo/redo steps.

• template<class Obj>
  Obj & undo_resolve(UndoDescriptor<Obj> &udo)
  Resolve an undo descriptor back to an object, see also undo_descriptor().

• template<class T>
  bool apply_idl_property(T &lvalue, const T &cvalue, const String &propname)
  Constrain and assign property value if it has changed, emit notification.

Detailed Description
Member Function Documentation

apply_idl_property()

template<class T>
bool Bse::ItemImpl::apply_idl_property (  
  T & lvalue,  
  const T & cvalue,  
  const String & propname ) [inline]
Constrain and assign property value if it has changed, emit notification.

push_property_undo()

void Bse::ItemImpl::push_property_undo (  
  const String & property_name )
Save the value of property_name onto the undo stack.

push_undo() [1/3]

template<typename ItemT, typename FuncArgs, typename... CallArgs>
void Bse::ItemImpl::push_undo (  
  const String & blurb,  
  ItemT & self,  
  Error(ItemT::*)(FuncArgs...) function,  
  CallArgs... args ) [inline]
Push an undo function onto the undo stack, the self argument to function must match this.

push_undo() [2/3]

template<typename ItemT, typename R, typename FuncArgs, typename... CallArgs>
void Bse::ItemImpl::push_undo (  
  const String & blurb,  
  ItemT & self,  
  R(ItemT::*)(FuncArgs...) function,  
  CallArgs... args ) [inline]
Push an undo function like push_undo(), but ignore the return value of function.
2.45 Bse::ItemSeq Struct Reference

A list of Item or derived objects.
import "bseapi.idl";

Detailed Description
A list of Item or derived objects.
The documentation for this struct was generated from the following file:

• bse/bseapi.idl

2.46 Bse::KeccakCryptoRng Class Reference

KeccakCryptoRng - A KeccakF1600 based cryptographic quality pseudo-random number generator.
#include <randomhash.hh>
Inheritance diagram for Bse::KeccakCryptoRng:

Public Member Functions

- **KeccakCryptoRng ()**
  
  *Initialize and seed the generator from a system specific nondeterministic random source.*

- template< class SeedSeq >
  
  **KeccakCryptoRng (SeedSeq &seed_sequence)**

  *Initialize and seed the generator from seed_sequence.*

Additional Inherited Members

Detailed Description

**KeccakCryptoRng** - A KeccakF1600 based cryptographic quality pseudo-random number generator. The quality of the generated pseudo random numbers is comparable to the hash output of SHAKE128.

Constructor & Destructor Documentation

**KeccakCryptoRng() [1/2]**

Bse::KeccakCryptoRng::KeccakCryptoRng ( ) [inline], [explicit]

*Initialize and seed the generator from a system specific nondeterministic random source.*
KeccakCryptoRng() [2/2]

```cpp
template<class SeedSeq >
Bse::KeccakCryptoRng::KeccakCryptoRng ( 
    SeedSeq & seed_sequence ) [inline], [explicit]
```

Initialize and seed the generator from `seed_sequence`.
The documentation for this class was generated from the following file:

- bse/randomhash.hh

2.47 Bse::Lib::KeccakF1600 Class Reference

The Keccak-f[1600] Permutation, see the Keccak specification Peeters und Assche (2011).

```cpp
#include <randomhash.hh>
```

Public Member Functions

- KeccakF1600 ()
  Zero the state.
- void reset ()
  Zero the state.
- void permute ( uint32_t n_rounds)
  Apply Keccak permutation with `n_rounds`.
- uint8_t & byte ( size_t state_index)

Detailed Description

The Keccak-f[1600] Permutation, see the Keccak specification Peeters und Assche (2011).

Constructor & Destructor Documentation

KeccakF1600()

```cpp
Bse::Lib::KeccakF1600::KeccakF1600 ( ) [explicit]
```

Zero the state.

Member Function Documentation

byte()

```cpp
uint8_t & Bse::Lib::KeccakF1600::byte ( 
    size_t state_index ) [inline]
```

Parameters

| state_index | Access byte 0..199 of the state. |

permute()

```cpp
void Bse::Lib::KeccakF1600::permute ( 
    uint32_t n_rounds )
```

Apply Keccak permutation with `n_rounds`. 
The Keccak-f[1600] permutation for up to 254 rounds, see Keccak11 Peeters und Assche (2011).

reset()

```cpp
void Bse::Lib::KeccakF1600::reset()
```

Zero the state.
The documentation for this class was generated from the following files:

- `bse/randomhash.hh`
- `bse/randomhash.cc`

2.48 Bse::KeccakFastRng Class Reference

KeccakFastRng - A KeccakF1600 based fast pseudo-random number generator.

```cpp
#include <randomhash.hh>
```

Inheritance diagram for Bse::KeccakFastRng:

```
Bse::KeccakRng
  + KeccakRng()
  + n_nums()
  + bit_capacity()
  + ~KeccakRng()
  + KeccakRng()
  + forget()
  + discard()
  + xor_seed()
  + seed()
  + seed()
  + seed()
  + seed()
  + seed()
  + seed()
  + seed()
  and 7 more...

Bse::KeccakFastRng
  + KeccakFastRng()
  + KeccakFastRng()
```

Public Member Functions

- **KeccakFastRng** ()
  
  *Initialize and seed the generator from a system specific nondeterministic random source.*

- template< class SeedSeq >
  
  **KeccakFastRng** (SeedSeq &seed_sequence)

  *Initialize and seed the generator from seed_sequence.*
Additional Inherited Members

Detailed Description

KeccakFastRng - A KeccakF1600 based fast pseudo-random number generator. This class tunes the KeccakF1600 algorithm for best performance in pseudo random number generation. Performance is improved while still retaining quality random number generation, according to the findings in section "4.1.1 Statistical tests" from http://keccak.noekeon.org/Keccak-reference-3.0.pdf.

Constructor & Destructor Documentation

KeccakFastRng() [1/2]

Bse::KeccakFastRng::KeccakFastRng ( ) [inline], [explicit]
Initialize and seed the generator from a system specific nondeterministic random source.

KeccakFastRng() [2/2]

template<class SeedSeq >
Bse::KeccakFastRng::KeccakFastRng ( SeedSeq & seed_sequence ) [inline], [explicit]
Initialize and seed the generator from seed_sequence. The documentation for this class was generated from the following file:

  * bse/randomhash.hh

2.49  Bse::KeccakGoodRng Class Reference

KeccakGoodRng - A KeccakF1600 based good quality pseudo-random number generator.
#include <randomhash.hh>
Inheritance diagram for Bse::KeccakGoodRng:

```
Bse::KeccakGoodRng
+ KeccakGoodRng()
+ KeccakGoodRng()

Bse::KeccakRng
+ KeccakRng()
+ n_nums()
+ bit_capacity()
+ ~KeccakRng()
+ KeccakRng()
+ forget()
+ discard()
+ xor_seed()
+ seed()
+ seed()
and 7 more...
```

Public Member Functions

- **KeccakGoodRng()**
  
  *Initialize and seed the generator from a system specific nondeterministic random source.*

- **template<class SeedSeq> KeccakGoodRng(SeedSeq &seed_sequence)**
  
  *Initialize and seed the generator from seed_sequence.*

Additional Inherited Members

Detailed Description

**KeccakGoodRng** - A KeccakF1600 based good quality pseudo-random number generator.

This class provides very good random numbers, using the KeccakF1600 algorithm without the extra security margins applied for SHA3 hash generation. This improves performance significantly without noticeably trading random number quality. For cryptography grade number generation *KeccakCryptoRng* should be used instead.

Constructor & Destructor Documentation

**KeccakGoodRng() [1/2]**

*Bse::KeccakGoodRng::KeccakGoodRng ( ) [inline], [explicit]*

*Initialize and seed the generator from a system specific nondeterministic random source.*
KeccakGoodRng() [2/2]

template<class SeedSeq >
Bse::KeccakGoodRng::KeccakGoodRng (  
    SeedSeq & seed_sequence ) [inline], [explicit]
Initialize and seed the generator from seed_sequence.
The documentation for this class was generated from the following file:

- bse/randomhash.hh

2.50  Bse::KeccakRng Class Reference

KeccakRng - A KeccakF1600 based pseudo-random number generator.
#include <randomhash.hh>
Inheritance diagram for Bse::KeccakRng:

Public Types

- typedef uint64_t result_type
  Integral type of the KeccakRng generator results.

Public Member Functions

- size_t n_nums () const
  Amount of 64 bit random numbers per generated block.
- size_t bit_capacity () const
  Amount of bits used to store hidden random number generator state.
- ~KeccakRng ()
  The destructor resets the generator state to avoid leaving memory trails.
- KeccakRng ( uint16_t hidden_state_capacity, uint16_t n_rounds)
  Create an unseeded Keccak PRNG with specific capacity and number of rounds, for experts only.
- void forget ()
  Discard 2^128 bits of the current generator state.
• void discard (unsigned long long count)
   Discard count consecutive random values.
• void xor_seed (const uint64_t *seeds, size_t n_seeds)
   Incorporate seed values into the current generator state.
• void seed (uint64_t seed_value=1)
   Reinitialize the generator state using a 64 bit seed value.
• void seed (const uint64_t *seeds, size_t n_seeds)
   Reinitialize the generator state using a nuber of 64 bit seeds.
• template<class SeedSeq>
  void seed (SeedSeq &seed_sequence)
   Seed the generator state from a seed sequence.
• void auto_seed ()
   Seed the generator from a system specific nondeterministic random source.
• uint64_t random ()
   Generate uniformly distributed 64 bit pseudo random number.
• result_type operator() ()
   Generate uniformly distributed 32 bit pseudo random number.
• template<typename RandomAccessIterator>
  void generate (RandomAccessIterator begin, RandomAccessIterator end)
   Fill the range [begin, end) with random unsigned integer values.
• result_type min () const
   Minimum of the result type, for uint64_t that is 0.
• result_type max () const
   Maximum of the result type, for uint64_t that is 18446744073709551615.

Friends

• bool operator== (const KeccakRng &lhs, const KeccakRng &rhs)
   Compare two generators for state equality.
• bool operator!= (const KeccakRng &lhs, const KeccakRng &rhs)
   Compare two generators for state inequality.
• template<typename CharT, typename Traits>
  std::basic_ostream<CharT, Traits> & operator<< ( std::basic_ostream<CharT, Traits> &os, const KeccakRng &self)
   Serialize generator state into an OStream.
• template<typename CharT, typename Traits>
  std::basic_istream<CharT, Traits> & operator>> ( std::basic_istream<CharT, Traits> &is, KeccakRng &self)
   Deserialize generator state from an IStream.

Detailed Description

KeccakRng - A KeccakF1600 based pseudo-random number generator. The permutation steps are derived from the Keccak specification Peeters und Assche (2011). For further details about this implementation, see also: http://testbit.org/keccak This class is primarily used to implement more fine tuned generators, such as: KeccakCryptoRng, KeccakGoodRng and KeccakFastRng.

Member Typedef Documentation

result_type

typedef uint64_t Bse::KeccakRng::result_type
Integral type of the KeccakRng generator results.
Constructor & Destructor Documentation

~KeccakRng()
Bse::KeccakRng::~KeccakRng()
The destructor resets the generator state to avoid leaving memory trails.

KeccakRng()
Bse::KeccakRng::KeccakRng
Create an unseeded Keccak PRNG with specific capacity and number of rounds, for experts only.

Member Function Documentation

auto_seed()
void Bse::KeccakRng::auto_seed()
Seed the generator from a system specific nondeterministic random source.

bit_capacity()
size_t Bse::KeccakRng::bit_capacity() const [inline]
Amount of bits used to store hidden random number generator state.

discard()
void Bse::KeccakRng::discard(unsigned long long count)
Discard count consecutive random values.
This function is slightly faster than calling operator() exactly count times.

forget()
void Bse::KeccakRng::forget()
Discard $2^{256}$ bits of the current generator state.
This makes it practically infeasible to guess previous generator states or deduce generated values from the past.
Use this for forward security Bellare und Yee (2001) of generated security tokens like session keys.

generate()
template<
typename RandomAccessIterator >
void Bse::KeccakRng::generate(RandomAccessIterator begin,
RandomAccessIterator end) [inline]
Fill the range [begin, end) with random unsigned integer values.

max()
result_type Bse::KeccakRng::max() const [inline]
Maximum of the result type, for uint64_t that is 18446744073709551615.
min()

```
result_type Bse::KeccakRng::min ( ) const [inline]
```
Minimum of the result type, for uint64_t that is 0.

n_nums()

```
size_t Bse::KeccakRng::n_nums ( ) const [inline]
```
Amount of 64 bit random numbers per generated block.

operator()

```
result_type Bse::KeccakRng::operator() ( ) [inline]
```
Generate uniformly distributed 32 bit pseudo random number.

random()

```
uint64_t Bse::KeccakRng::random ( ) [inline]
```
Generate uniformly distributed 64 bit pseudo random number.
A new block permutation is carried out every n_nums() calls, see also xor_seed().

seed() [1/3]

```
void Bse::KeccakRng::seed (  
    uint64_t seed_value = 1 ) [inline]
```
Reinitialize the generator state using a 64 bit seed_value.

seed() [2/3]

```
void Bse::KeccakRng::seed (  
    const uint64_t *seeds,  
    size_t n_seeds ) [inline]
```
Reinitialize the generator state using a nuber of 64 bit seeds.

seed() [3/3]

```
template<class SeedSeq >
void Bse::KeccakRng::seed (  
    SeedSeq & seed_sequence ) [inline]
```
Seed the generator state from a seed_sequence.

xor_seed()

```
void Bse::KeccakRng::xor_seed (  
    const uint64_t *seeds,  
    size_t n_seeds )
```
Incorporate seed_values into the current generator state.
A block permutation to advance the generator state is carried out per n_nums() seed values. After calling this function, generating the next n_nums() random values will not need to block for a new permutation.

**Friends And Related Function Documentation**
operator"!=

bool operator!= (  
    const KeccakRng & lhs,  
    const KeccakRng & rhs ) [friend]

Compare two generators for state inequality.

operator<<

template<typename CharT , typename Traits >  
std::basic_ostream<CharT, Traits>& operator<<(  
    std::basic_ostream<CharT, Traits> & os,  
    const KeccakRng & self ) [friend]

Serialize generator state into an OStream.

operator==

bool operator== (  
    const KeccakRng & lhs,  
    const KeccakRng & rhs ) [friend]

Compare two generators for state equality.

operator>>

template<typename CharT , typename Traits >  
std::basic_istream<CharT, Traits>& operator>>(  
    std::basic_istream<CharT, Traits> & is,  
    KeccakRng & self ) [friend]

Deserialize generator state from an IStream.

The documentation for this class was generated from the following files:

- bse/randomhash.hh
- bse/randomhash.cc

2.51 Bse::LegacyObject Interface Reference

Base type for all legacy objects, derived from struct BseObject.

import"bseapi.idl";
Inheritance diagram for Bse::LegacyObject:

```
Bse::Object
+ set_prop()
+ get_prop()
+ find_prop()
+ list_props()
+ notify()
```

```
Bse::LegacyObject
+ uname
+ debug_name()
+ unique_id()
+ proxy_id()
+ find_typedata()
```

```
Bse::Item
+ icon
+ seqid
+ use()
+ unuse()
+ set_name()
+ common_ancestor()
+ check_is_a()
+ group_undo()
+ ungroup_undo()
+ get_project()
+ get_parent()
+ get_seqid()
+ and 11 more...
```

```
Bse::LegacyObjectIface
```

```
Bse::Object
+ set_prop()
+ get_prop()
+ find_prop()
+ list_props()
+ notify()
```

```
Bse::LegacyObject
+ uname
+ debug_name()
+ unique_id()
+ proxy_id()
+ find_typedata()
```

```
Bse::Item
+ icon
+ seqid
+ ... find_typedata()
+ as_bse_object()
```

```
Bse::LegacyObjectImpl
+ LegacyObjectImpl()
+ ~LegacyObjectImpl()
+ operator BseObject *()
+ debug_name()
+ unique_id()
+ uname()
+ uname()
+ proxy_id()
+ find_typedata()
+ as_bse_object()
```

**Public Member Functions**

- String **debug_name** ()
  
  *Object name useful for debugging output.*

- int32 **unique_id** ()
  
  *Generate a unique stable ID for this object, similar to a hash of its address.*

- int64 **proxy_id** ()
  
  *Helper ID for the old SfIGlue interface.*

- StringSeq **find_typedata** (String **type_name**)
Query key=value meta info for type type_name.

Detailed Description
Base type for all legacy objects, derived from struct BseObject.

Events:
- **notify** - A notification event for property changes, the event field `detail` contains the property name.

Member Function Documentation

debug_name()

```cpp
String Bse::LegacyObject::debug_name ( )
```
Object name useful for debugging output.

find_typedata()

```cpp
StringSeq Bse::LegacyObject::find_typedata ( String type_name )
```
Query key=value meta info for type type_name.

proxy_id()

```cpp
int64 Bse::LegacyObject::proxy_id ( )
```
Helper ID for the old SfiGlue interface.

unique_id()

```cpp
int32 Bse::LegacyObject::unique_id ( )
```
Generate a unique stable ID for this object, similar to a hash of its address. The documentation for this interface was generated from the following file:

- bse/bseapi.idl

2.52 Bse::LegacyObjectIface Class Reference

IDL interface class for Bse::LegacyObject.

```cpp
#include <bseapi_interfaces.hh>
```
Inheritance diagram for Bse::LegacyObjectIface:

Additional Inherited Members

Detailed Description

IDL interface class for Bse::LegacyObject.
The documentation for this class was generated from the following file:

- bse/bseapi_interfaces.hh
2.53 Bse::LegacyObjectImpl Class Reference

Inheritance diagram for Bse::LegacyObjectImpl:

Additional Inherited Members

Detailed Description

The documentation for this class was generated from the following files:

- bse/bseobject.hh
- bse/bseobject.cc

2.54 Bse::MidiNotifier Interface Reference

Interface for MIDI event notification.
import "bseapi.idl";
Inheritance diagram for Bse::MidiNotifier:

Additional Inherited Members

Detailed Description

Interface for MIDI event notification.
The documentation for this interface was generated from the following file:

- bse/bseapi.idl
2.55  Bse::MidiNotifierIface Class Reference

IDL interface class for Bse::MidiNotifier.
#include <bseapi_interfaces.hh>
Inheritance diagram for Bse::MidiNotifierIface:

![Inheritance Diagram]

**Additional Inherited Members**

**Detailed Description**

IDL interface class for Bse::MidiNotifier.
The documentation for this class was generated from the following file:

- bse/bseapi_interfaces.hh
### 2.56 Bse::MidiNotifierImpl Class Reference

Inheritance diagram for Bse::MidiNotifierImpl:

```
Bse::MidiNotifier

+ MidiNotifierImpl()
Bse::ItemImpl

+ ItemImpl()
+ parent()
+ use()
+ unuse()
+ set_name()
+ editable_property()
+ icon()
+ icon()
+ common_ancestor()
+ check_is_a()
and 25 more...
Bse::MidiNotifierIface

Bse::MidiNotifier

Additional Inherited Members

Detailed Description

The documentation for this class was generated from the following files:

- bse/bsemidinotifier.hh
- bse/bsemidinotifier.cc

### 2.57 Bse::MidiSynth Interface Reference

Interface for MIDI synthesis networks.

```cpp
import"bseapi.idl";
```
Inheritance diagram for Bse::MidiSynth:

Additional Inherited Members

Detailed Description

Interface for MIDI synthesis networks.
The documentation for this interface was generated from the following file:

- bse/bseapi.idl
IDL interface class for Bse::MidiSynth.
#include <bseapi_interfaces.hh>
Inheritance diagram for Bse::MidiSynthIface:

![Inheritance Diagram](image)

Additional Inherited Members

Detailed Description

IDL interface class for Bse::MidiSynth.
The documentation for this class was generated from the following file:

- bse/bseapi_interfaces.hh
2.59  Bse::MidiSynthImpl Class Reference

Inheritance diagram for Bse::MidiSynthImpl:

Additional Inherited Members

Detailed Description
The documentation for this class was generated from the following files:

- bse/bsemidisynth.hh
- bse/bsemidisynth.cc

2.60  Bse::Module Class Reference

Interface for the encapsulation of audio processors.
import"bseengine.node.hh";
Inheritance diagram for Bse::Module:

Public Member Functions

- **String get_module_type ()**
  
  *Retrieve type of module to be created.*

- **ModuleTypeInfo module_type_info ()**

  *Describe this module type.*
Detailed Description

Interface for the encapsulation of audio processors. DSP Engine Module.

Member Function Documentation

get_module_type()

String Bse::Module::get_module_type()  
Retrieve type of module to be created.

module_type_info()

ModuleTypeInfo Bse::Module::module_type_info()  
Describe this module type.

The documentation for this class was generated from the following files:

- bse/bseapi.idl
- bse/bseenginenode.hh
- bse/bseengine.cc

2.61 Bse::ModuleIface Class Reference

IDL interface class for Bse::Module.
#include <bseapi_interfaces.hh>
Inheritance diagram for Bse::ModuleIface:

Additional Inherited Members

Detailed Description

IDL interface class for Bse::Module.
The documentation for this class was generated from the following file:

- bse/bseapi_interfaces.hh
2.62  Bse::ModuleImpl Class Reference

Inheritance diagram for Bse::ModuleImpl:

Additional Inherited Members

Detailed Description

The documentation for this class was generated from the following files:

- bse/module.hh
- bse/module.cc
2.63 Bse::ModuleTypeInfo Struct Reference

Info for module types.
import"bseapi.idl";

Detailed Description
Info for module types.
The documentation for this struct was generated from the following file:

- bse/bseapi.idl

2.64 Bse::NoteDescription Struct Reference

A note description provides all needed details about a specific note.
import"bseapi.idl";

Detailed Description
A note description provides all needed details about a specific note.
The documentation for this struct was generated from the following file:

- bse/bseapi.idl

2.65 Bse::Object Interface Reference

Base type for all new style C++ objects.
import"bseapi.idl";
Inheritance diagram for Bse::Object:
Public Member Functions

- bool set_prop (String name, Any value)
  Set dynamic property name to value.
- Any get_prop (String name)
  Get value of dynamic property name.
- StringSeq find_prop (String name)
  Query key=value meta info for dynamic property name.
- StringSeq list_props ()
  List names of all dynamic properties.
- void notify (String detail)
  Emit notification event for object state or property changes.

Detailed Description

Base type for all new style C++ objects.

Member Function Documentation

find_prop()

StringSeq Bse::Object::find_prop (String name)
Query key=value meta info for dynamic property name.

get_prop()

Any Bse::Object::get_prop (String name)
Get value of dynamic property name.

list_props()

StringSeq Bse::Object::list_props ()
List names of all dynamic properties.

notify()

void Bse::Object::notify (String detail)
Emit notification event for object state or property changes.

set_prop()

bool Bse::Object::set_prop (String name, Any value)
Set dynamic property name to value.
The documentation for this interface was generated from the following file:

- bse/bseapi.idl
2.66 Bse::ObjectIface Class Reference

IDL interface class for Bse::Object.
#include <bseapi_interfaces.hh>
Inheritance diagram for Bse::ObjectIface:

```
Bse::Object
  + set_prop()
  + get_prop()
  + find_prop()
  + list_props()
  + notify()

Bse::ObjectIface

Bse::ObjectImpl
  + ObjectImpl()
  + set_prop()
  + get_prop()
  + find_prop()
  + list_props()
  + notify()
  + emit_event()
  + __execution_context_mt__()
  + __attach__()

Bse::Object
  + set_prop()
  + get_prop()
  + find_prop()
  + list_props()
  + notify()
```

Additional Inherited Members

Detailed Description

IDL interface class for Bse::Object.
The documentation for this class was generated from the following file:

* bse/bseapi_interfaces.hh
2.67  Bse::ObjectImpl Class Reference

Inheritance diagram for Bse::ObjectImpl:

Additional Inherited Members

Detailed Description

The documentation for this class was generated from the following files:

- bse/object.hh
- bse/object.cc

2.68  Bse::Part Interface Reference

Data interface for containment of piano notes and MIDI effects.

import "bseapi.idl";
Inheritance diagram for Bse::Part:

Public Member Functions

- **PartControlSeq list_selected_controls** (MidiSignal control_type)
  
  *List all currently selected control events of a specific type.*

- **PartControlSeq list_controls** (int32 tick, int32 duration, MidiSignal control_type)
  
  *List all control events within a tick range.*

- **PartControlSeq get_channel_controls** (int32 channel, int32 tick, int32 duration, MidiSignal control_type)
  
  *Retrieve all control events of a specific type within range of a channel.*

- **PartControlSeq get_controls** (int32 tick, MidiSignal control_type)
Retrieve all control events of a specific type at specified tick.

- **SongTiming get_timing (int32 tick)**
  Retrieve song timing information at a specific tick.

- **int32 get_max_note ()**
  Retrieve the maximum note supported in this part.

- **int32 get_min_note ()**
  Retrieve the minimum note supported in this part.

- **int32 get_last_tick ()**
  Retrieve the maximum tick any control or note plus duration spans.

- **Error change_control (int32 id, int32 tick, MidiSignal control_type, float64 value)**
  Change an existing control event within a part.

- **Error change_note (int32 id, int32 tick, int32 duration, int32 note, int32 fine_tune, float64 velocity)**
  Change an existing note within a part.

- **Error delete_event (int32 id)**
  Delete an existing event from a part.

- **int32 insert_control (int32 tick, MidiSignal control_type, float64 value)**
  Insert a new control event into a part.

- **int32 insert_note (int32 channel, int32 tick, int32 duration, int32 note, int32 fine_tune, float64 velocity)**
  Insert a new note into a part.

- **int32 insert_note_auto (int32 tick, int32 duration, int32 note, int32 fine_tune, float64 velocity)**
  Insert a new note into a part with automatic channel selection.

- **bool is_event_selected (int32 id)**
  Check whether an event is selected.

- **void queue_controls (int32 tick, int32 duration)**
  Queue updates for all control events and notes starting within the given range.

- **void queue_notes (int32 tick, int32 duration, int32 min_note, int32 max_note)**
  Queue updates for all notes starting within the given rectangle.

- **void select_notes_exclusive (int32 tick, int32 duration, int32 min_note, int32 max_note)**
  Select all notes within rectangle and deselect all others.

- **void select_notes_exclusive (int32 tick, int32 duration, int32 min_note, int32 max_note)**
  Select all control events within range and deselect all others.

- **void select_notes (int32 tick, int32 duration, int32 min_note, int32 max_note)**
  Select all notes within rectangle.

- **void select_event (int32 id)**
  Select an existing event.

- **void select_controls (int32 tick, int32 duration, MidiSignal control_type)**
  Select all control events within range.

- **void deselect_notes (int32 tick, int32 duration, int32 min_note, int32 max_note)**
  Deselect all notes within rectangle.

- **void deselect_notes (int32 tick, int32 duration, int32 min_note, int32 max_note)**
  Deselect all control events within range.

- **void deselect_event (int32 id)**
  Deselect an existing event.

- **void deselect_controls (int32 tick, int32 duration, MidiSignal control_type)**
  Deselect all controls within given range.

- **PartNoteSeq list_notes_crossing (int32 tick, int32 duration)**
  List all notes within or crossing a tick range.

- **PartNoteSeq list_notes_within (int32 channel, int32 tick, int32 duration)**
  List all notes within a tick range.

- **PartNoteSeq list_selected_notes ()**
  List all currently selected notes.

- **PartNoteSeq check_overlap (int32 tick, int32 duration, int32 note)**
  Check whether a note would overlap with neighbours.

- **PartNoteSeq get_notes (int32 tick, int32 note)**
  Retrieve all notes at a specific frequency or crossing a tick.

- **PartLinkSeq list_links ()**
  List all places where parts are used (linked) from tracks, sorted by tick.
Detailed Description
Data interface for containment of piano notes and MIDI effects.

Member Function Documentation

change_control()
Error Bse::Part::change_control (  
   int32 id,  
   int32 tick,  
   MidiSignal control_type,  
   float64 value )  
Change an existing control event within a part.

change_note()
Error Bse::Part::change_note (  
   int32 id,  
   int32 tick,  
   int32 duration,  
   int32 note,  
   int32 fine_tune,  
   float64 velocity )  
Change an existing note within a part.

check_overlap()
PartNoteSeq Bse::Part::check_overlap (  
   int32 tick,  
   int32 duration,  
   int32 note )  
Check whether a note would overlap with neighbours.

delete_event()
Error Bse::Part::delete_event (  
   int32 id )  
Delete an existing event from a part.

deselect_controls()
void Bse::Part::deselect_controls (  
   int32 tick,  
   int32 duration,  
   MidiSignal control_type )  
Deselect all controls within given range.

deselect_event()
void Bse::Part::deselect_event (  
   int32 id )  
Deselect an existing event.
deselect_notes()
void Bse::Part::deselect_notes (  
    int32 tick,  
    int32 duration,  
    int32 min_note,  
    int32 max_note )
Deselect all notes within rectangle.

get_channel_controls()
PartControlSeq Bse::Part::get_channel_controls (  
    int32 channel,  
    int32 tick,  
    int32 duration,  
    MidiSignal control_type )
Retrieve all control events of a specific type within range of a channel.

get_controls()
PartControlSeq Bse::Part::get_controls (  
    int32 tick,  
    MidiSignal control_type )
Retrieve all control events of a specific type at specified tick.

get_last_tick()
int32 Bse::Part::get_last_tick ( )
Retrieve the maximum tick any control or note plus duration spans.

get_max_note()
int32 Bse::Part::get_max_note ( )
Retrieve the maximum note supported in this part.

get_min_note()
int32 Bse::Part::get_min_note ( )
Retrieve the minimum note supported in this part.

get_notes()
PartNoteSeq Bse::Part::get_notes (  
    int32 tick,  
    int32 note )
Retrieve all notes at a specific frequency or crossing a tick.

get_timing()
SongTiming Bse::Part::get_timing (  
    int32 tick )
Retrieve song timing information at a specific tick.
insert_control()

```cpp
int32 Bse::Part::insert_control (  
    int32 tick,  
    MidiSignal control_type,  
    float64 value )
```
Insert a new control event into a part.

**insert_note()**

```cpp
int32 Bse::Part::insert_note (  
    int32 channel,  
    int32 tick,  
    int32 duration,  
    int32 note,  
    int32 fine_tune,  
    float64 velocity )
```
Insert a new note into a part.

**insert_note_auto()**

```cpp
int32 Bse::Part::insert_note_auto (  
    int32 tick,  
    int32 duration,  
    int32 note,  
    int32 fine_tune,  
    float64 velocity )
```
Insert a new note into a part with automatic channel selection.

**is_event_selected()**

```cpp
bool Bse::Part::is_event_selected (  
    int32 id )
```
Check whether an event is selected.

**list_controls()**

```cpp
PartControlSeq Bse::Part::list_controls (  
    int32 tick,  
    int32 duration,  
    MidiSignal control_type )
```
List all control events within a tick range.

**list_links()**

```cpp
PartLinkSeq Bse::Part::list_links (  
)
```
List all places where parts are used (linked) from tracks, sorted by tick.

**list_notes_crossing()**

```cpp
PartNoteSeq Bse::Part::list_notes_crossing (  
    int32 tick,  
    int32 duration )
```
List all notes within or crossing a tick range.
list_notes_within()

```cpp
PartNoteSeq Bse::Part::list_notes_within (  
    int32 channel,  
    int32 tick,  
    int32 duration )
```
List all notes within a tick range.

list_selected_controls()

```cpp
PartControlSeq Bse::Part::list_selected_controls (  
    MidiSignal control_type )
```
List all currently selected control events of a specific type.

list_selected_notes()

```cpp
PartNoteSeq Bse::Part::list_selected_notes ( )
```
List all currently selected notes.

queue_controls()

```cpp
void Bse::Part::queue_controls (  
    int32 tick,  
    int32 duration )
```
Queue updates for all control events and notes starting within the given range.

queue_notes()

```cpp
void Bse::Part::queue_notes (  
    int32 tick,  
    int32 duration,  
    int32 min_note,  
    int32 max_note )
```
Queue updates for all notes starting within the given rectangle.

select_controls()

```cpp
void Bse::Part::select_controls (  
    int32 tick,  
    int32 duration,  
    MidiSignal control_type )
```
Select all control events within range.

select_controls_exclusive()

```cpp
void Bse::Part::select_controls_exclusive (  
    int32 tick,  
    int32 duration,  
    MidiSignal control_type )
```
Select all control events within range and deselect all others.
select_event()

void Bse::Part::select_event ( int32 id )
Select an existing event.

select_notes()

void Bse::Part::select_notes ( int32 tick, int32 duration, int32 min_note, int32 max_note )
Select all notes within rectangle.

select_notes_exclusive()

void Bse::Part::select_notes_exclusive ( int32 tick, int32 duration, int32 min_note, int32 max_note )
Select all notes within rectangle and deselect all others.
The documentation for this interface was generated from the following file:
• bse/bseapi.idl

2.69 Bse::PartControl Struct Reference

Part specific control event representation.
import"bseapi.idl";

Detailed Description
Part specific control event representation.
The documentation for this struct was generated from the following file:
• bse/bseapi.idl

2.70 Bse::PartControlSeq Struct Reference

A list of part control events.
import"bseapi.idl";

Detailed Description
A list of part control events.
The documentation for this struct was generated from the following file:
• bse/bseapi.idl
2.71 Bse::PartIface Class Reference

IDL interface class for Bse::Part.
#include <bseapi_interfaces.hh>

Inheritance diagram for Bse::PartIface:

![Inheritance diagram]

**Additional Inherited Members**

**Detailed Description**

IDL interface class for Bse::Part.
The documentation for this class was generated from the following file:
2.72 Bse::PartImpl Class Reference

Inheritance diagram for Bse::PartImpl:

Additional Inherited Members

Detailed Description

The documentation for this class was generated from the following files:

- bse/bseapi_interfaces.hh
2.73 Bse::PartLink Struct Reference

Record representing the use of a Part within a Track at a specific position.
import "bseapi.idl";

Detailed Description
Record representing the use of a Part within a Track at a specific position.
The documentation for this struct was generated from the following file:
• bse/bseapi.idl

2.74 Bse::PartLinkSeq Struct Reference

Sequence of PartLink records.
import "bseapi.idl";

Detailed Description
Sequence of PartLink records.
The documentation for this struct was generated from the following file:
• bse/bseapi.idl

2.75 Bse::PartNote Struct Reference

Part specific note event representation.
import "bseapi.idl";

Detailed Description
Part specific note event representation.
The documentation for this struct was generated from the following file:
• bse/bseapi.idl

2.76 Bse::PartNoteSeq Struct Reference

A list of part note events.
import "bseapi.idl";

Detailed Description
A list of part note events.
The documentation for this struct was generated from the following file:
• bse/bseapi.idl

2.77 Bse::PartSeq Struct Reference

A list of Part or derived types.
import "bseapi.idl";
Detailed Description

A list of Part or derived types.
The documentation for this struct was generated from the following file:

- bse/bseapi.idl

2.78 Bse::Pcg32Rng Class Reference

Pcg32Rng is a permutating linear congruential PRNG.

#include <randomhash.hh>

Public Member Functions

- template<class SeedSeq >
  Pcg32Rng (SeedSeq &seed_sequence)
  Initialize and seed from seed_sequence.
- Pcg32Rng ( uint64_t offset, uint64_t struct )
  Initialize and seed by seeking to position offset within stream sequence.
- Pcg32Rng ()
  Initialize and seed the generator from a system specific nondeterministic random source.
- void auto_seed ()
  Seed the generator from a system specific nondeterministic random source.
- void seed ( uint64_t offset, uint64_t struct )
  Seed by seeking to position offset within stream sequence.
- template<class SeedSeq >
  void seed (SeedSeq &seed_sequence)
  Seed the generator state from a seed_sequence.
- uint32_t random ()
  Generate uniformly distributed 32 bit pseudo random number.

Detailed Description

Pcg32Rng is a permutating linear congruential PRNG.
At the core, this pseudorandom number generator uses the well known linear congruential generator:
\( 6364136223846793005 \times \text{accumulator} + 1442695040888963407 \mod 2^{64} \). See also TAOCP by D. E. Knuth, section 3.3.4, table 1, line 26. For good statistical performance, the output function of the permuted congruential generator family is used as described on http://www.pcg-random.org/. Period length for this generator is \( 2^{64} \), the specified seed offset chooses the position of the generator and the seed sequence parameter can be used to choose from \( 2^{63} \) distinct random sequences.

Constructor & Destructor Documentation

Pcg32Rng() [1/3]

template<class SeedSeq >
Bse::Pcg32Rng::Pcg32Rng (SeedSeq & seed_sequence ) [inline], [explicit]
Initialize and seed from seed_sequence.
Bse::Pcg32Rng() [2/3]
Bse::Pcg32Rng::Pcg32Rng (  
    uint64_t offset,  
    uint64_t struct ) [explicit]
Initialize and seed by seeking to position offset within stream sequence.

Bse::Pcg32Rng() [3/3]
Bse::Pcg32Rng::Pcg32Rng ( ) [explicit]
Initialize and seed the generator from a system specific nondeterministic random source.

Member Function Documentation

auto_seed()

void Bse::Pcg32Rng::auto_seed ( )
Seed the generator from a system specific nondeterministic random source.

random()

uint32_t Bse::Pcg32Rng::random ( ) [inline]
Generate uniformly distributed 32 bit pseudo random number.

seed() [1/2]

void Bse::Pcg32Rng::seed (  
    uint64_t offset,  
    uint64_t struct )
Seed by seeking to position offset within stream sequence.

seed() [2/2]

template<class SeedSeq >
void Bse::Pcg32Rng::seed (  
    SeedSeq & seed_sequence ) [inline]
Seed the generator state from a seed_sequence.
The documentation for this class was generated from the following files:

- bse/randomhash.hh
- bse/randomhash.cc

2.79 Bse::PcmWriter Interface Reference

Interface for writing PCM wave data.
import"bseapi.idl";
Inheritance diagram for Bse::PcmWriter:

### Additional Inherited Members

### Detailed Description

Interface for writing PCM wave data.

The documentation for this interface was generated from the following file:

- bse/bseapi.idl
2.80 Bse::PcmWriterIface Class Reference

IDL interface class for Bse::PcmWriter.
#include <bseapi_interfaces.hh>
Inheritance diagram for Bse::PcmWriterIface:

```
+ Bse::PcmWriter

Bse::PcmWriterIface

+ Bse::PcmWriterImpl
  + PcmWriterImpl()
  + trigger_tick()

```

Additional Inherited Members

Detailed Description

IDL interface class for Bse::PcmWriter.
The documentation for this class was generated from the following file:

- bse/bseapi_interfaces.hh
2.81  Bse::PcmWriterImpl Class Reference

Inheritance diagram for Bse::PcmWriterImpl:

![Inheritance Diagram]

**Additional Inherited Members**

**Detailed Description**

The documentation for this class was generated from the following files:

- bse/bsepcmwriter.hh
- bse/bsepcmwriter.cc

2.82  Bse::PixelSeq Struct Reference

Representation of an image pixel sequence in ARGB format.

```plaintext
import "bseapi.idl";
```

**Detailed Description**

Representation of an image pixel sequence in ARGB format.
2.83 Bse::ProbeFeatures Struct Reference

Bits representing a selection of probe sample data features.

```cpp
import "bseapi.idl";
```

**Detailed Description**

Bits representing a selection of probe sample data features.

The documentation for this struct was generated from the following file:

- `bse/bseapi.idl`

2.84 Bse::Project Interface Reference

Projects support loading, saving, playback and act as containers for all other sound objects.

```cpp
import "bseapi.idl";
```
Inheritance diagram for Bse::Project:

Public Member Functions

- **ProjectState get_state ()**
  
  Retrieve the current project activation/playback state.

- **void change_name (String name)**
  
  Change a project name without recording undo steps.

- **Error play ()**
  
  Activate a project and start project playback (an already playing project is first halted).

- **Error activate ()**
Activate a project, precondition to start playback.

- `bool can_play()`
  * Check whether project playback would make sense.

- `bool is_playing()`
  * Check whether a project is currently playing (song sequencing).

- `bool is_active()`
  * Check whether a project is active (currently synthesizing).

- `void start_playback()`
  * Start playback in an activated project.

- `void stop_playback()`
  * Stop project playback.

- `void deactivate()`
  * Deactivate the project, automatically stop playback.

- `void stop()`
  * Stop project playback and deactivate project.

- `void auto_deactivate(int32 msec_delay)`
  * Automatically deactivate a few milliseconds after playback stopped.

- `int32 undo_depth()`
  * Check whether a project can perform undo steps.

- `void undo()`
  * Undo a previous operation in a project.

- `int32 redo_depth()`
  * Get the number of times redo can be called on the project.

- `void redo()`
  * Redo a previously undone operation in a project.

- `void clear_undo()`
  * Delete all recorded undo or redo steps.

- `void clean_dirty()`
  * Clear a project’s dirty flags.

- `bool is_dirty()`
  * Check whether a project needs saving.

- `SuperSeq get_supers()`
  * Retrieve all Super type objects of this project.

- `Error store(String file_name, bool self_contained)`
  * Save project to file.

- `Error store_bse(Container super, String file_name, bool self_contained)`
  * Save effect or instrument.

- `Song create_song(String name)`
  * Create a song for this project.

- `WaveRepo get_wave_repo()`
  * Retrieve the project’s unique wave repository.

- `SoundFontRepo get_sound_font_repo()`
  * Retrieve the project’s unique sound font repository.

- `CSynth create_csynth(String name)`
  * Create a synthesizer network for this project.

- `MidiSynth create_midi_synth(String name)`
  * Create a MIDI synthesizer network for this project.

- `MidiNotifier get_midi_notifier()`
  * Retrieve the project’s midi notifier object.

- `void remove_snet(SNet snet)`
  * Remove an existing synthesizer network from this project.

- `Error restore_from_file(String file_name)`
Load a project from file.

- void inject_midi_control (int32 midi_channel, int32 midi_control, float64 control_value)

  Inject a MIDI control event into the project's MIDI receiver.

- Error import_midi_file (String file_name)

  Import a song from a MIDI file.

Public Attributes

- group State

  List unamed paths for all items of a specified type within a project.

Detailed Description

Projects support loading, saving, playback and act as containers for all other sound objects.

Member Function Documentation

activate()

Error Bse::Project::activate ( )
Activate a project, precondition to start playback.

auto_deactivate()

void Bse::Project::auto_deactivate ( int32 msec_delay )
Automatically deactivate a few milliseconds after playback stopped.

can_play()

bool Bse::Project::can_play ( )
Check whether project playback would makes sense.

change_name()

void Bse::Project::change_name ( String name )
Change a project name without recording undo steps.

clean_dirty()

void Bse::Project::clean_dirty ( )
Clear a project's dirty flags.

clear_undo()

void Bse::Project::clear_undo ( )
Delete all recorded undo or redo steps.
create_csynth()

```cpp
CSynth Bse::Project::create_csynth (String name)
```

Create a synthesizer network for this project.

create_midi_synth()

```cpp
MidiSynth Bse::Project::create_midi_synth (String name)
```

Create a MIDI synthesizer network for this project.

create_song()

```cpp
Song Bse::Project::create_song (String name)
```

Create a song for this project.

deactivate()

```cpp
void Bse::Project::deactivate ()
```

Deactivate the project, automatically stop playback.

get_midi_notifier()

```cpp
MidiNotifier Bse::Project::get_midiNotifier ( )
```

Retrieve the project's midi notifier object.

get_sound_font_repo()

```cpp
SoundFontRepo Bse::Project::get_sound_font_repo ( )
```

Retrieve the project's unique sound font repository.

get_state()

```cpp
ProjectState Bse::Project::get_state ( )
```

Retrieve the current project activation/playback state.

get_supers()

```cpp
SuperSeq Bse::Project::get_supers ( )
```

Retrieve all Super type objects of this project.

get_wave_repo()

```cpp
WaveRepo Bse::Project::get_wave_repo ( )
```

Retrieve the project's unique wave repository.
import_midi_file()

Error Bse::Project::import_midi_file (String file_name)
Import a song from a MIDI file.

inject_midi_control()

void Bse::Project::inject_midi_control (int32 midi_channel, int32 midi_control, float64 control_value)
Inject a MIDI control event into the project's MIDI receiver.

is_active()

bool Bse::Project::is_active ()
Check whether a project is active (currently synthesizing).

is_dirty()

bool Bse::Project::is_dirty ()
Check whether a project needs saving.

is_playing()

bool Bse::Project::is_playing ()
Check whether a project is currently playing (song sequencing).

play()

Error Bse::Project::play ()
Activate a project and start project playback (an already playing project is first halted).

redo()

void Bse::Project::redo ()
Redo a previously undone operation in a project.

redo_depth()

int32 Bse::Project::redo_depth ()
Get the number of times redo can be called on the project.

remove_snet()

void Bse::Project::remove_snet (SNet snet)
Remove an existing synthesizer network from this project.
**restore_from_file()**

Error Bse::Project::restore_from_file (  
    String file_name  )
Load a project from file.

**start_playback()**

void Bse::Project::start_playback ( )
Start playback in an activated project.

**stop()**

void Bse::Project::stop ( )
Stop project playback and deactivate project.

**stop_playback()**

void Bse::Project::stop_playback ( )
Stop project playback.

**store()**

Error Bse::Project::store (  
    String file_name,  
    bool self_contained )
Save project to file.

**store_bse()**

Error Bse::Project::store_bse (  
    Container super,  
    String file_name,  
    bool self_contained )
Save effect or instrument.

**undo()**

void Bse::Project::undo ( )
Undo a previous operation in a project.

**undo_depth()**

int32 Bse::Project::undo_depth ( )
Check whether a project can perform undo steps.

**Member Data Documentation**
2.85 Bse::ProjectIface Class Reference

IDL interface class for Bse::Project.

#include <bseapi_interfaces.hh>

State

group Bse::Project::State
List uname paths for all items of a specified type within a project.
By their uname paths, items are uniquely identifiable within a project. Retrieve all items of a specific type
within a project with matching uname. Save super objects of a project into a BSE file. If no Super is specified,
the project itself is stored. The references to other objects (e.g. samples) can be stored by reference (self_contained = false) or embedded in the output file (self_contained = true).
The documentation for this interface was generated from the following file:

  • bse/bseapi.idl
Inheritance diagram for Bse::ProjectIface:

Additional Inherited Members

Detailed Description

IDL interface class for Bse::Project.

The documentation for this class was generated from the following file:

- bse/bseapi_interfaces.hh
2.86 Bse::ProjectImpl Class Reference

Inheritance diagram for Bse::ProjectImpl:

Additional Inherited Members

Detailed Description

The documentation for this class was generated from the following files:

- bse/bseproject.hh
- bse/bseproject.cc
2.87  Bse::PropertyCandidates Struct Reference

A list of items suitable to set as a specific property value.

```cpp
import "bseapi.idl";
```

**Detailed Description**

A list of items suitable to set as a specific property value.

The documentation for this struct was generated from the following file:

- `bse/bseapi.idl`

2.88  Bse::Xms::SerializationNode::QueuedArgs Struct Reference

Helper for deferred xml_reflink() calls.

```cpp
#include <serializable.hh>
```

**Detailed Description**

Helper for deferred xml_reflink() calls.

The documentation for this struct was generated from the following file:

- `bse/serializable.hh`

2.89  Bse::Xms::Reflink Class Reference

Representation for an object reference between SerializableInterface objects.

```cpp
#include <serializable.hh>
```

**Detailed Description**

Representation for an object reference between SerializableInterface objects.

The documentation for this class was generated from the following files:

- `bse/serializable.hh`
- `bse/serializable.cc`

2.90  Bse::Resampler2 Class Reference

Interface for factor 2 resampling classes.

```cpp
#include <bseresampler.hh>
```

**Public Member Functions**

- **Resampler2** (Mode mode, Precision precision, bool use_sse_if_available = true)
  
  *creates a resampler instance fulfilling a given specification*

- void **process_block** (const float *input, uint n_input_samples, float *output)
  
  *resample a data block*

- uint **order** () const
  
  *return FIR filter order*

- **delay** () const
  
  *Return the delay introduced by the resampler.*

- void **reset** ()
  
  *clear internal history, reset resampler state to zero values*

- bool **sse_enabled** () const
  
  *return whether the resampler is using sse optimized code*
Static Public Member Functions

- static bool sse_available ()
  returns true if an optimized SSE version of the Resampler is available
- static bool test_filter_impl (bool verbose)
  test internal filter implementation
- static Precision find_precision_for_bits (uint bits)
  finds a precision which is appropriate for at least the specified number of bits
- static const char * precision_name (Precision precision)
  returns a human-readable name for a given precision

Detailed Description

Interface for factor 2 resampling classes.

Constructor & Destructor Documentation

Resampler2()

Resampler2::Resampler2 (Mode mode,
  Precision precision,
  bool use_sse_if_available = true )
creates a resampler instance fulfilling a given specification

Member Function Documentation

delay()

double Bse::Resampler2::delay ( ) const [inline]
Return the delay introduced by the resampler. This delay is guaranteed to be \( \geq 0.0 \), and for factor 2 resampling always a multiple of 0.5 (1.0 for upsampling). The return value can also be thought of as index into the output signal, where the first input sample can be found. Beware of fractional delays, for instance for downsampling, a \( \text{delay}() \) of 10.5 means that the first input sample would be found by interpolating output[10] and output[11], and the second input sample equates output[11].

find_precision_for_bits()

Resampler2::Precision Resampler2::find_precision_for_bits (uint bits) [static]
finds a precision which is appropriate for at least the specified number of bits

order()

uint Bse::Resampler2::order ( ) const [inline]
return FIR filter order
precision_name()

const char * Resampler2::precision_name (  
    Precision precision ) [static]

returns a human-readable name for a given precision

process_block()

void Bse::Resampler2::process_block (  
    const float * input,  
    uint n_input_samples,  
    float * output ) [inline]

resample a data block

reset()

void Bse::Resampler2::reset ( ) [inline]

clear internal history, reset resampler state to zero values

sse_available()

bool Resampler2::sse_available ( ) [static]

returns true if an optimized SSE version of the Resampler is available

sse_enabled()

bool Bse::Resampler2::sse_enabled ( ) const [inline]

return whether the resampler is using sse optimized code

test_filter_impl()

bool Resampler2::test_filter_impl (  
    bool verbose ) [static]

test internal filter implementation

The documentation for this class was generated from the following files:

• bse/bseresampler.hh
• bse/bseresampler.cc

2.91 Bse::SampleFileInfo Struct Reference

Structure containing meta data for multi wave samples.

import"bseapi.idl";

Detailed Description

Structure containing meta data for multi wave samples.

The documentation for this struct was generated from the following file:

• bse/bseapi.idl
2.92 Bse::Lib::ScopedLocale Class Reference

Class to push a specific locale_t for the scope of its lifetime.

#include <formatter.hh>

Inheritance diagram for Bse::Lib::ScopedLocale:

![Inheritance Diagram](image)

Detailed Description

Class to push a specific locale_t for the scope of its lifetime.
The documentation for this class was generated from the following files:

- bse/formatter.hh
- bse/formatter.cc

2.93 Bse::Lib::ScopedPosixLocale Class Reference

Class to push the POSIX/C locale_t (UTF-8) for the scope of its lifetime.

#include <formatter.hh>
Inheritance diagram for Bse::Lib::ScopedPosixLocale:

```
Bse::Lib::ScopedLocale
+ ~ScopedLocale()
Bse::Lib::ScopedPosixLocale
+ ScopedPosixLocale()
+ posix_locale()
```

**Static Public Member Functions**

- static locale_t **posix_locale** ()
  
  *Retrieve the (UTF-8) POSIX/C locale_t.*

**Detailed Description**

Class to push the POSIX/C locale_t (UTF-8) for the scope of its lifetime.

**Member Function Documentation**

**posix_locale()**

```cpp
locale_t Bse::Lib::ScopedPosixLocale::posix_locale ( ) [static]
```

*Retrieve the (UTF-8) POSIX/C locale_t.*

The documentation for this class was generated from the following files:

- bse/formatter.hh
- bse/formatter.cc

**2.94 Bse::Sequencer Class Reference**

Note and MIDI sequencer.

```cpp
#include <bsesequencer.hh>
```

**Detailed Description**

Note and MIDI sequencer.

The sequencer processes notes from parts and MIDI input and generates events for the synthesis engine.

The documentation for this class was generated from the following files:

- bse/bsesequencer.hh
- bse/bsesequencer.cc
2.95 Bse::Xms::SerializableInterface Class Reference

Interface for serializable objects with Reflink support.
#include <serializable.hh>
Inheritance diagram for Bse::Xms::SerializableInterface:

![Inheritance diagram for Bse::Xms::SerializableInterface]

**Detailed Description**

Interface for serializable objects with Reflink support.
The documentation for this class was generated from the following files:

- bse/serializable.hh
- bse/serializable.cc

2.96 Bse::Xms::SerializationField Class Reference

Reference to a storage attribute (or child node) for serialization.
#include <serializable.hh>

**Public Member Functions**

- template<typename T, typename E = void>
  void operator & (T &value)
  *Serialization operator.*
- SerializationField & node ()
  *Force storage into child node (not attribute)*
- bool as_node () const
  *Retrieve node() flag.*
- SerializationField & hex ()
**Detailed Description**

Reference to a storage attribute (or child node) for serialization.

**Member Function Documentation**

**as_hex()**

bool Bse::Xms::SerializationField::as_hex() const

Retrive hex() flag.

**as_node()**

bool Bse::Xms::SerializationField::as_node() const

Retrive node() flag.

**attribute()**

String Bse::Xms::SerializationField::attribute() const

Associated attribute (or child node) name for serialization.

**hex()**

SerializationField & Bse::Xms::SerializationField::hex()

Hint for unsigned storage as hexadecimal.

**node()**

SerializationField & Bse::Xms::SerializationField::node()

Force storage into child node (not attribute)

**operator &()**

```cpp
template<typename T, typename E = void>
void Bse::Xms::SerializationField::operator& (T & value)
```

Serialization operator.
2.97 Bse::Xms::SerializationNode Class Reference

Representation of a storage node for serialization via `save()` and `load()`

#include <serializable.hh>

Classes

- struct QueuedArgs
  
  Helper for deferred xml_reflink() calls.

Public Member Functions

- SerializationNode()
  
  Create a SerializationNode de-/serialization.

- String name() const
  
  Tag name of this storage node.

- operator bool () const noexcept
  
  Check if this exists as deserialization node.

- SerializationNode create_child(const String &childname)
  
  Create a child node for nested hierarchies.

- SerializationNode first_child(const String &childname)
  
  Access the first stored child node.

- Children children(const String &childname)
  
  List all children by name.

- SerializationField get(const String &attrib)
  
  Refer to a serialization field by name.

- SerializationField operator[](const String &attrib)
  
  Convenience operator alias for get()

- bool has(const String &attrib) const
  
  Check for attribute or child presence.

- bool loading(const String &attrib) const
  
  Combines in_load() with has() check.

- bool in_load() const
  
  Return true during deserialization.

- bool in_save() const
  
  Return true during serialization.

- bool with_default() const
  
  Always true during in_load(), toggled during in_save()

- void with_default(bool dflt)
  
  Toggle with_default() during in_save()

- template<typename T >
  
  Reflink & reflink(T * &serializable_ptr)
  
  Wrap serializable object pointers.

- template<typename T >
  
  void save_under(const String &tag, T &object)
**Serializing object into a child node.**

- Template:
  ```cpp
template<typename T>
void load (T &object)
```

**Deserializing object via xml::serialize() and xml::reflink()**

- Template:
  ```cpp
template<typename T>
void save (T &object)
```

**Serialization object via xml::serialize() and xml::reflink()**

- Function:
  ```cpp
Bse::Error parse_xml (const String &root, const String &xmltext, String *ep = NULL)
```
  Parses `xmltext`, returns error location and message in `ep`.

- Function:
  ```cpp
String write_xml (const String &root)
```
  Generates XML with top level tag `root` from this `SerializationNode`.

**Static Public Attributes**

- Variable:
  ```cpp
static constexpr const char *const null_id
```
  String representing a serialized `nullptr`.

**Detailed Description**

Representation of a storage node for serialization via `save()` and `load()`.

**Constructor & Destructor Documentation**

- Function:
  ```cpp
SerializationNode()
```
  Creates a `SerializationNode` de-/serialization.

**Member Function Documentation**

- Function:
  ```cpp
SerializationNode::Children Bse::Xms::SerializationNode::children (const String &childname)
```
  List all children by name.

- Function:
  ```cpp
SerializationNode Bse::Xms::SerializationNode::create_child (const String &childname)
```
  Creates a child node for nested hierarchies.

- Function:
  ```cpp
SerializationNode Bse::Xms::SerializationNode::first_child (const String &childname)
```
  Accesses the first stored child node.
get()

```cpp
SerializationField Bse::Xms::SerializationNode::get (  
    const String & attrib
)
```
Refer to a serialization field by name.

has()

```cpp
bool Bse::Xms::SerializationNode::has (  
    const String & attrib
) const
```
Check for attribute or child presence.

in_load()

```cpp
bool Bse::Xms::SerializationNode::in_load ( ) const
```
Return true during deserialization.

in_save()

```cpp
bool Bse::Xms::SerializationNode::in_save ( ) const
```
Return true during serialization.

load()

```cpp
template<
    typename T
>
void Bse::Xms::SerializationNode::load (  
    T & object
)
```
Deserialize object via xml_serialize() and xml_reflink()

loading()

```cpp
bool Bse::Xms::SerializationNode::loading (  
    const String & attrib
) const
```
Combines in_load() with has() check.

name()

```cpp
String Bse::Xms::SerializationNode::name ( ) const
```
Tag name of this storage node.

operator bool()

```cpp
Bse::Xms::SerializationNode::operator bool ( ) const [explicit], [noexcept]
```
Check if this exists as deserialization node.

operator[]()

```cpp
SerializationField Bse::Xms::SerializationNode::operator[] (  
    const String & attrib
)
```
Convenience operator alias for get()
parse_xml()

Bse::Error Bse::Xms::SerializationNode::parse_xml (const String & root, const String & xmltext,
String * ep = NULL )

Parse xmlstr, returns error location and message in ep.

relink()

template<typename T >
Reflink & Bse::Xms::SerializationNode::relink (T * & serializable_ptr )

Wrap serializable object pointers.

save()

template<typename T >
void Bse::Xms::SerializationNode::save (T & object )

Serialize object via xml_serialize() and xml_relink()

save_under()

template<typename T >
void Bse::Xms::SerializationNode::save_under (const String & tag, T & object )

Serialize object into a child node.

with_default() [1/2]

bool Bse::Xms::SerializationNode::with_default () const
Always true during in_load(), toggled during in_save()

with_default() [2/2]

void Bse::Xms::SerializationNode::with_default (bool dflt )
Toggle with_default() during in_save()

write_xml()

String Bse::Xms::SerializationNode::write_xml (const String & root )

Generate XML with toplevel tag root from this SerializationNode.

Member Data Documentation
null_id

constexpr const char* const Bse::Xms::SerializationNode::null_id [static]
String representing a serialized nullptr

The documentation for this class was generated from the following files:
- bse/serializable.hh
- bse/serializable.cc

2.98  **Bse::Server Interface Reference**

Main Bse remote origin object.

import "bseapi.idl";
Inheritance diagram for Bse::Server:

Public Member Functions

- void send_user_message (UserMessage umsg)
  Send a user messages from BSE.
- LegacyObject from_proxy (int64 proxyid)
  Find an Object from its associated BseObject proxy id.
- bool engine_active ()
  Retrieve DSP engine activateion state, see also: "enginechange" Event.
- String get_mp3_version ()
Retrieve BSE MP3 handler version.

- String get_vorbis_version ()
  Retrieve BSE Vorbis handler version.

- String get_ladspa_path ()
  Retrieve ladspa search path.

- String get_plugin_path ()
  Retrieve plugin search path.

- String get_instrument_path ()
  Retrieve instrument search path.

- String get_sample_path ()
  Retrieve sample search path.

- String get_effect_path ()
  Retrieve effect search path.

- String get_demo_path ()
  Retrieve demo search path.

- String get_custom_instrument_dir ()
  Retrieve user specific instruments directory.

- String get_custom_effect_dir ()
  Retrieve user specific effects directory.

- String get_version ()
  Retrieve BSE version.

- void purge_stalecachedirs ()
  Purge stale directories from past runtimes.

- void register_ladspa_plugins ()
  Register LADSPA (Linux Audio Developer's Simple Plugin API) plugins asynchronously, DEPRECATED.

- void register_core_plugins ()
  Register core plugins asynchronously, DEPRECATED.

- void load_assets ()
  Register plugins and scripts immediately.

- bool can_load (String file_name)
  Check whether a loader can be found for a wave file.

- void start_recording (String wave_file, float64 n_seconds)
  Start recording to a WAV file.

- Project create_project (String project_name)
  Create a new project (name is modified to be unique if necessary).

- Project last_project ()
  Retrieve the last created project.

- void destroy_project (Project project)
  Destroy a previously created new project.

- AuxDataSeq list_module_types ()
  A list of Source type names for create_source().

- AuxData find_module_type (String module_type)
  Retrieve info about a Source type names.

- Icon module_type_icon (String module_type)
  Retrieve the icon associated with a module type.

- SampleFileInfo sample_file_info (String file_name)
  Load sample file info from file.

- void broadcast_shm_fragments (ShmFragmentSeq plan, int32 interval_ms)
  Broadcast shared memory fragments to the current Jsonipc connection.

- SharedMemory get_shared_memory ()
  Retrieve global SharedMemory information.

- Configuration get_config_defaults ()
Retrieve Bse::Configuration setting defaults.

• void set_config (Configuration configuration)

Assign updated Bse::Configuration settings.

• Configuration get_config ()

Retrieve Bse::Configuration settings.

• bool locked_config ()
  Returns whether Configuration is in use and locked against modifications.

• DriverEntrySeq list_pcm_drivers ()
  List available drivers for PCM input/output handling.

• DriverEntrySeq list_midi_drivers ()
  List available drivers for MIDI input/output handling.

• NoteDescription note_describe (MusicalTuning musical_tuning, int32 note, int32 fine_tune)
  Describe a note, providing information about its octave, semitone, frequency, etc.

• NoteDescription note_construct (MusicalTuning musical_tuning, int32 semitone, int32 octave, int32 fine_tune)
  Describe a note, given its semitone, octave and fine tune.

• NoteDescription note_from_string (MusicalTuning musical_tuning, String name)
  Describe a note, given its name and octave offset.

• int32 note_from_freq (MusicalTuning musical_tuning, float64 frequency)
  Retrieve the note of a certain frequency.

• float64 note_to_freq (MusicalTuning musical_tuning, int32 note, int32 fine_tune)
  Retrieve the frequency of a certain note.

• CategorySeq category_match_typed (String pattern, String type_name)
  List BSE categories according to a pattern and type match.

• CategorySeq category_match (String pattern)
  List BSE categories according to a pattern match.

• int64 tick_stamp_from_systime (int64 systime_usecs)
  Helper for Wave PCM positioning.

• int32 test_counter_inc_fetch ()
  Testing, increment and return the resulting test counter value.

Detailed Description

Main Bse remote origin object.
The Bse::Server object controls the main BSE thread and keeps track of all objects used in the BSE context.

Events:

• enginechange - A notification event for DSP engine changes, the event field active contains the activation state.

Member Function Documentation

broadcast_shm_fragments()

void Bse::Server::broadcast_shm_fragments ( 
  ShmFragmentSeq plan, 
  int32 interval_ms )

Broadcast shared memory fragments to the current Jsonipc connection.
can_load()

bool Bse::Server::can_load ( String file_name )
Check whether a loader can be found for a wave file.

category_match()

CategorySeq Bse::Server::category_match ( String pattern )
List BSE categories according to a pattern match.

category_match_typed()

CategorySeq Bse::Server::category_match_typed ( String pattern, String type_name )
List BSE categories according to a pattern and type match.

create_project()

Project Bse::Server::create_project ( String project_name )
Create a new project (name is modified to be unique if necessary.

destroy_project()

void Bse::Server::destroy_project ( Project project )
Destroy a previously created new project.

disable_proxy()

bool Bse::Server::disable_proxy ( )

engine_active()

bool Bse::Server::engine_active ( )
Retrieve DSP engine activateion state, see also: "enginechange" Event.

find_module_type()

AuxData Bse::Server::find_module_type ( String module_type )
Retrieve info about a Source type names.

from_proxy()

LegacyObject Bse::Server::from_proxy ( int64 proxyid )
Find an Object from its associated BseObject proxy id.

get_config()

Configuration Bse::Server::get_config ( )
Retrieve Bse::Configuration settings.
get_config_defaults()
Configuration Bse::Server::get_config_defaults ( )
Retrieve Bse::Configuration setting defaults.

get_custom_effect_dir()
String Bse::Server::get_custom_effect_dir ( )
Retrieve user specific effects directory.

get_custom_instrument_dir()
String Bse::Server::get_custom_instrument_dir ( )
Retrieve user specific instruments directory.

get_demo_path()
String Bse::Server::get_demo_path ( )
Retrieve demo search path.

get_effect_path()
String Bse::Server::get_effect_path ( )
Retrieve effect search path.

get_instrument_path()
String Bse::Server::get_instrument_path ( )
Retrieve instrument search path.

get_ladspa_path()
String Bse::Server::get_ladspa_path ( )
Retrieve ladspa search path.

get_mp3_version()
String Bse::Server::get_mp3_version ( )
Retrieve BSE MP3 handler version.

get_plugin_path()
String Bse::Server::get_plugin_path ( )
Retrieve plugin search path.

get_sample_path()
String Bse::Server::get_sample_path ( )
Retrieve sample search path.
get_shared_memory()

SharedMemory Bse::Server::get_shared_memory ( )
Retrieve global SharedMemory information.

get_version()

String Bse::Server::get_version ( )
Retrieve BSE version.

get_vorbis_version()

String Bse::Server::get_vorbis_version ( )
Retrieve BSE Vorbis handler version.

last_project()

Project Bse::Server::last_project ( )
Retrieve the last created project.

list_midi_drivers()

DriverEntrySeq Bse::Server::list_midi_drivers ( )
List available drivers for MIDI input/output handling.

list_module_types()

AuxDataSeq Bse::Server::list_module_types ( )
A list of Source type names for create_source().

list_pcm_drivers()

DriverEntrySeq Bse::Server::list_pcm_drivers ( )
List available drivers for PCM input/output handling.

load_assets()

void Bse::Server::load_assets ( )
Register plugins and scripts immediately.

locked_config()

bool Bse::Server::locked_config ( )
Returns whether Configuration is in use and locked against modifications.

module_type_icon()

Icon Bse::Server::module_type_icon ( 
  String module_type )
Retrieve the icon associated with a module type.
note_construct()

NoteDescription Bse::Server::note_construct (  
    MusicalTuning musical_tuning,  
    int32 semitone,  
    int32 octave,  
    int32 fine_tune )
Describe a note, given its semitone, octave and fine tune.

note_describe()

NoteDescription Bse::Server::note_describe (  
    MusicalTuning musical_tuning,  
    int32 note,  
    int32 fine_tune )
Describe a note, providing information about its octave, semitone, frequency, etc.

note_from_freq()

int32 Bse::Server::note_from_freq (  
    MusicalTuning musical_tuning,  
    float64 frequency )
Retrieve the note of a certain frequency.

note_from_string()

NoteDescription Bse::Server::note_from_string (  
    MusicalTuning musical_tuning,  
    String name )
Describe a note, given its name and octave offset.

note_to_freq()

float64 Bse::Server::note_to_freq (  
    MusicalTuning musical_tuning,  
    int32 note,  
    int32 fine_tune )
Retrieve the frequency of a certain note.

purge_stale_cachedirs()

void Bse::Server::purge_stale_cachedirs ( )
Purge stale directories from past runtimes.

register_core_plugins()

void Bse::Server::register_core_plugins ( )
Register core plugins asynchronously, DEPRECATED.

register_ladspa_plugins()

void Bse::Server::register_ladspa_plugins ( )
Register LADSPA (Linux Audio Developer's Simple Plugin API) plugins asynchronously, DEPRECATED.
sample_file_info()

SampleFileInfo Bse::Server::sample_file_info (  
  String file_name  )  
Load sample file info from file.

send_user_message()

void Bse::Server::send_user_message (  
  UserMessage umsg  )  
Send a user messages from BSE.

set_config()

void Bse::Server::set_config (  
  Configuration configuration  )  
Assign updated Bse::Configuration settings.

start_recording()

void Bse::Server::start_recording (  
  String wave_file,  
  float64 n_seconds  )  
Start recording to a WAV file.

test_counter_inc_fetch()

int32 Bse::Server::test_counter_inc_fetch ( )  
Testing, increment and return the resulting test counter value.

tick_stamp_from_systime()

int64 Bse::Server::tick_stamp_from_systime (  
  int64 systime_usecs  )  
Helper for Wave PCM positioning.
The documentation for this interface was generated from the following file:

• bse/bseapi.idl

2.99  Bse::ServerIface Class Reference

IDL interface class for Bse::Server.
#include <bseapi_interfaces.hh>
Inheritance diagram for Bse::ServerIface:

```
Bse::Server
+ Misc
+ Recording
  + send_user_message()
  + from_proxy()
  + engine_active()
  + get_mp3_version()
  + get_vorbis_version()
  + get_ladspa_path()
  + get_plugin_path()
  + get_instrument_path()
  + get_sample_path()
  + get_effect_path()
  and 38 more...

Bse::ServerIface

Bse::ServerImpl

+ enginechange()
+ allocate_shared_block()
+ release_shared_block()
+ set_ipc_handler()
+ get_ipc_handler()
+ open_midi_driver()
+ close_midi_driver()
+ pcm_driver()
+ open_pcm_driver()
+ require_pcm_input()
  and 54 more...
+ register_source_module()
+ instance()
```

**Additional Inherited Members**

**Detailed Description**

IDL interface class for Bse::Server.
The documentation for this class was generated from the following file:

- bse/bseapi_interfaces.hh
2.100 Bse::ServerImpl Class Reference

Inheritance diagram for Bse::ServerImpl:

```
Bse::Server
+ Misc
+ Recording
+ send_user_message()
+ from_proxy()
+ engine_active()
+ get_mp3_version()
+ get_vorbis_version()
+ get_ladspa_path()
+ get_plugin_path()
+ get_instrument_path()
+ get_sample_path()
+ get_effect_path()
and 38 more...
```

```
Bse::ServerIface
+ Misc
+ Recording
+ send_user_message()
+ from_proxy()
+ engine_active()
+ get_mp3_version()
+ get_vorbis_version()
+ get_ladspa_path()
+ get_plugin_path()
+ get_instrument_path()
+ get_sample_path()
+ get_effect_path()
and 38 more...
```

```
Bse::ContainerImpl
+ ContainerImpl()
+ lookup_item()
+ get_item()
+ list_children()
```

Static Public Member Functions

- static void register_source_module (const String &type, const String &title, const String &tags, const uint8 *pixstream)

  Register a synthesis module type at program startup.
Additional Inherited Members

Detailed Description

Member Function Documentation

register_source_module()

```cpp
void Bse::ServerImpl::register_source_module (const String & type,
   const String & title,
   const String & tags,
   const uint8 * pixstream ) [static]
```

Register a synthesis module type at program startup.
The documentation for this class was generated from the following files:

- bse/beserver.hh
- bse/beserver.cc

## 2.101 SfiRecFields Struct Reference

Pointer sized integer object handle.

```cpp
#include <sfitypes.hh>
```

### Detailed Description

Pointer sized integer object handle.
The documentation for this struct was generated from the following file:

- bse/sfitypes.hh

## 2.102 Bse::SHA3_224 Struct Reference

SHA3_224 - 224 Bit digest generation.

```cpp
#include <randomhash.hh>
```

### Public Member Functions

- **SHA3_224 ()**
  
  *Create context to calculate a 224 bit SHA3 hash digest.*

- **void reset ()**
  
  *Reset state to feed and retrieve a new hash value.*

- **void update (const uint8_t *data, size_t length)**
  
  *Feed data to be hashed.*

- **void digest (uint8_t hashvalue[28])**
  
  *Retrieve the resulting hash value.*

### Detailed Description

SHA3_224 - 224 Bit digest generation.

This class implements the SHA3 hash function to create 224 Bit digests, see FIPS 202 Division (2014).

### Constructor & Destructor Documentation
SHA3_224()

Bse::SHA3_224::SHA3_224 ( )
Create context to calculate a 224 bit SHA3 hash digest.

Member Function Documentation

digest()

void Bse::SHA3_224::digest ( 
    uint8_t hashvalue[28] )
Retrieve the resulting hash value.

reset()

void Bse::SHA3_224::reset ( )
Reset state to feed and retrieve a new hash value.

update()

void Bse::SHA3_224::update ( 
    const uint8_t ∗data, 
    size_t length )
Feed data to be hashed.
The documentation for this struct was generated from the following files:

• bse/randomhash.hh
• bse/randomhash.cc

2.103  Bse::SHA3_256 Struct Reference

SHA3_256 - 256 Bit digest generation.
#include <randomhash.hh>

Public Member Functions

• SHA3_256 ()
    Create context to calculate a 256 bit SHA3 hash digest.
• void reset ()
    Reset state to feed and retrieve a new hash value.
• void update (const uint8_t ∗data, size_t length)
    Feed data to be hashed.
• void digest ( uint8_t hashvalue[32])
    Retrieve the resulting hash value.

Detailed Description

SHA3_256 - 256 Bit digest generation.
This class implements the SHA3 hash function to create 256 Bit digests, see FIPS 202 Division (2014).

Constructor & Destructor Documentation
SHA3_256()

Bse::SHA3_256::SHA3_256()
Create context to calculate a 256 bit SHA3 hash digest.

Member Function Documentation

digest()

void Bse::SHA3_256::digest {
    uint8_t hashvalue[32]
}
Retrieve the resulting hash value.

reset()

void Bse::SHA3_256::reset ()
Reset state to feed and retrieve a new hash value.

update()

void Bse::SHA3_256::update {
    const uint8_t *data,
    size_t length
}
Feed data to be hashed.
The documentation for this struct was generated from the following files:

- bse/randomhash.hh
- bse/randomhash.cc

2.104  Bse::SHA3_384 Struct Reference

SHA3_384 - 384 Bit digest generation.
#include <randomhash.hh>

Public Member Functions

- SHA3_384 ()
  Create context to calculate a 384 bit SHA3 hash digest.
- void reset ()
  Reset state to feed and retrieve a new hash value.
- void update (const uint8_t *data, size_t length)
  Feed data to be hashed.
- void digest (uint8_t hashvalue[48])
  Retrieve the resulting hash value.

Detailed Description

SHA3_384 - 384 Bit digest generation.
This class implements the SHA3 hash function to create 384 Bit digests, see FIPS 202 Division (2014).

Constructor & Destructor Documentation
SHA3_384()

Bse::SHA3_384::SHA3_384()
Create context to calculate a 384 bit SHA3 hash digest.

Member Function Documentation

digest()

void Bse::SHA3_384::digest (  
    uint8_t hashvalue[48] )
Retrieve the resulting hash value.

reset()

void Bse::SHA3_384::reset (  )
Reset state to feed and retrieve a new hash value.

update()

void Bse::SHA3_384::update (  
    const uint8_t *data,  
    size_t length )
Feed data to be hashed.
The documentation for this struct was generated from the following files:

• bse/randomhash.hh
• bse/randomhash.cc

2.105 Bse::SHA3_512 Struct Reference

SHA3_512 - 512 Bit digest generation.
#include <randomhash.hh>

Public Member Functions

• SHA3_512 ()
    Create context to calculate a 512 bit SHA3 hash digest.
• void reset ()
    Reset state to feed and retrieve a new hash value.
• void update (const uint8_t *data, size_t length)
    Feed data to be hashed.
• void digest (uint8_t hashvalue[64])
    Retrieve the resulting hash value.

Detailed Description

SHA3_512 - 512 Bit digest generation.
This class implements the SHA3 hash function to create 512 Bit digests, see FIPS 202 Division (2014).
SHA3_512()

Bse::SHA3_512::SHA3_512 ( )
Create context to calculate a 512 bit SHA3 hash digest.

Member Function Documentation

digest()

void Bse::SHA3_512::digest ( 
    uint8_t hashvalue[64] )
Retrieves the resulting hash value.

reset()

void Bse::SHA3_512::reset ( )
Reset state to feed and retrieve a new hash value.

update()

void Bse::SHA3_512::update ( 
    const uint8_t *data, 
    size_t length )
Feed data to be hashed.
The documentation for this struct was generated from the following files:
- bse/randomhash.hh
- bse/randomhash.cc

2.106  Bse::SHAKE128 Struct Reference

SHAKE128 - 128 Bit extendable output digest generation.
#include <randomhash.hh>

Public Member Functions

- SHAKE128 ( )
  Create context to calculate an unbounded SHAKE128 hash digest.
- void reset ( )
  Reset state to feed and retrieve a new hash value.
- void update (const uint8_t *data, size_t length)
  Feed data to be hashed.
- void squeeze_digest (uint8_t *hashvalues, size_t n)
  Retrieve an arbitrary number of hash value bytes.

Detailed Description

SHAKE128 - 128 Bit extendable output digest generation.
This class implements the SHA3 extendable output hash function with 128 bit security strength, see FIPS 202 Division (2014).

Constructor & Destructor Documentation
SHAKE128()

Bse::SHAKE128::SHAKE128 ( )
Create context to calculate an unbounded SHAKE128 hash digest.

Member Function Documentation

reset()

void Bse::SHAKE128::reset ( )
Reset state to feed and retrieve a new hash value.

squeeze_digest()

void Bse::SHAKE128::squeeze_digest ( uint8_t ∗ hashvalues, size_t n )
Retrieve an arbitrary number of hash value bytes.

update()

void Bse::SHAKE128::update ( const uint8_t ∗ data, size_t length )
Feed data to be hashed.
The documentation for this struct was generated from the following files:
• bse/randomhash.hh
• bse/randomhash.cc

2.107 Bse::SHAKE256 Struct Reference

SHAKE256 - 256 Bit extendable output digest generation.
#include <randomhash.hh>

Public Member Functions

• SHAKE256 ( )
  Create context to calculate an unbounded SHAKE256 hash digest.
• void reset ( )
  Reset state to feed and retrieve a new hash value.
• void update ( const uint8_t ∗ data, size_t length )
  Feed data to be hashed.
• void squeeze_digest ( uint8_t ∗ hashvalues, size_t n )
  Retrieve an arbitrary number of hash value bytes.

Detailed Description

SHAKE256 - 256 Bit extendable output digest generation.
This class implements the SHA3 extendable output hash function with 256 bit security strength, see FIPS 202 Division (2014).

Constructor & Destructor Documentation
SHAKE256()

Bse::SHAKE256::SHAKE256 ( )
Create context to calculate an unbounded SHAKE256 hash digest.

Member Function Documentation

reset()

void Bse::SHAKE256::reset ( )
Reset state to feed and retrieve a new hash value.

squeeze_digest()

void Bse::SHAKE256::squeeze_digest ( 
    uint8_t * hashvalues,
    size_t n )
Retrieve an arbitrary number of hash value bytes.

update()

void Bse::SHAKE256::update ( 
    const uint8_t * data,
    size_t length )
Feed data to be hashed.

The documentation for this struct was generated from the following files:

• bse/randomhash.hh
• bse/randomhash.cc

2.108  Bse::SharedMemory Struct Reference

Descriptor for a shared memory region.
import"bseapi.idl";

Public Attributes

• int64 shm_creator
  IPC id of the shared memory creator process.
• int64 shm_start
  Shared memory area location.
• int64 shm_length
  Shared memory area length in bytes.

Detailed Description

Descriptor for a shared memory region.

Member Data Documentation


**shm_creator**

`int64 Bse::SharedMemory::shm_creator`

IPC id of the shared memory creator process.

**shm_length**

`int64 Bse::SharedMemory::shm_length`

Shared memory area length in bytes.

**shm_start**

`int64 Bse::SharedMemory::shm_start`

Shared memory area location.

The documentation for this struct was generated from the following file:

* bse/bseapi.idl

### 2.109 Bse::ShmFragment Struct Reference

Fragment description for interesting bits of shared memory.

import"bseapi.idl";

**Detailed Description**

Fragment description for interesting bits of shared memory.

The documentation for this struct was generated from the following file:

* bse/bseapi.idl

### 2.110 Bse::ShmFragmentSeq Struct Reference

Collection of shared memory fragments.

import"bseapi.idl";

**Detailed Description**

Collection of shared memory fragments.

The documentation for this struct was generated from the following file:

* bse/bseapi.idl

### 2.111 Bse::SignalMonitor Interface Reference

Interface for monitoring output signals.

import"bseapi.idl";
Inheritance diagram for Bse::SignalMonitor:

Public Member Functions

- **Source get_osource ()**
  
  *Retrieve output module the SignalMonitor is connected to.*

- **int32 get_ochannel ()**
  
  *Retrieve output channel the SignalMonitor is connected to.*

- **int64 get_mix_freq ()**
  
  *Mix frequency at which monitor values are calculated.*

- **int64 get_frame_duration ()**
Frame duration in \(\mu\)seconds for the calculation of monitor values.

- \texttt{int64 get\_shm\_offset (MonitorField fld)}
  
  Offset into shared memory for MonitorField values of ochannel.

- \texttt{void set\_probe\_features (ProbeFeatures pf)}
  
  Configure probe features.

- \texttt{ProbeFeatures get\_probe\_features ()}
  
  Get configured probe features.

**Detailed Description**

Interface for monitoring output signals.

**Member Function Documentation**

**get\_frame\_duration()**

\texttt{int64 Bse::SignalMonitor::get\_frame\_duration ()}

Frame duration in \(\mu\)seconds for the calculation of monitor values.

**get\_mix\_freq()**

\texttt{int64 Bse::SignalMonitor::get\_mix\_freq ()}

Mix frequency at which monitor values are calculated.

**get\_ochannel()**

\texttt{int32 Bse::SignalMonitor::get\_ochannel ()}

Retrieve output channel the \texttt{SignalMonitor} is connected to.

**get\_osource()**

\texttt{Source Bse::SignalMonitor::get\_osource ()}

Retrieve output module the \texttt{SignalMonitor} is connected to.

**get\_probe\_features()**

\texttt{ProbeFeatures Bse::SignalMonitor::get\_probe\_features ()}

Get configured probe features.

**get\_shm\_offset()**

\texttt{int64 Bse::SignalMonitor::get\_shm\_offset (MonitorField fld)}

Offset into shared memory for MonitorField values of ochannel.

**set\_probe\_features()**

\texttt{void Bse::SignalMonitor::set\_probe\_features (ProbeFeatures pf)}

Configure probe features.

The documentation for this interface was generated from the following file:

- \texttt{bse/bseapi.idl}
2.112 Bse::SignalMonitorIface Class Reference

IDL interface class for Bse::SignalMonitor.
#include <bseapi_interfaces.hh>

Inheritance diagram for Bse::SignalMonitorIface:

```
+ get_osource()
+ get_ochannel()
+ get_mix_freq()
+ get_frame_duration()
+ get_shm_offset()
+ set_probe_features()
+ get_probe_features()
```

```
+ SignalMonitorImpl()
+ get_osource()
+ get_shm_offset()
+ get_ochannel()
+ get_mix_freq()
+ get_frame_duration()
+ set_probe_features()
+ get_probe_features()
```

**Additional Inherited Members**

**Detailed Description**

IDL interface class for Bse::SignalMonitor.
The documentation for this class was generated from the following file:

- bse/bseapi_interfaces.hh
2.113 **Bse::SignalMonitorImpl Class Reference**

Inheritance diagram for Bse::SignalMonitorImpl:

![Inheritance Diagram]

**Additional Inherited Members**

**Detailed Description**

The documentation for this class was generated from the following files:

- bse/monitor.hh
- bse/monitor.cc

2.114 **Bse::SNet Interface Reference**

Base interface type for all kinds of synthesis networks.

```python
import "bseapi.idl";
```
Inheritance diagram for Bse::SNet:

Public Member Functions

- **bool supports_user_synths()**
  
  *Check whether users may edit synthesis modules of this network.*

- **Error can_create_source(String module_type)**
  
  *Check whether inserting a new module into a synthesis network is possible.*

- **Source create_source(String module_type)**
  
  *Insert a new module into a synthesis network.*

- **Error remove_source(Source module)**
  
  *Remove an existing module from its synthesis network.*

Detailed Description

Base interface type for all kinds of synthesis networks.

Member Function Documentation
can_create_source()

Error Bse::SNet::can_create_source ( String module_type )
Check whether inserting a new module into a synthesis network is possible.

can_create_source()

create_source()

Source Bse::SNet::create_source ( String module_type )
Insert a new module into a synthesis network.

create_source()

remove_source()

Error Bse::SNet::remove_source ( Source module )
Remove an existing module from its synthesis network.

remove_source()

supports_user_synths()

bool Bse::SNet::supports_user_synths ( )
Check whether users may edit synthesis modules of this network.
The documentation for this interface was generated from the following file:

• bse/bseapi.idl

2.115 Bse::SNetIface Class Reference

IDL interface class for Bse::SNet.
#include <bseapi_interfaces.hh>
Inheritance diagram for Bse::SNetIface:

```
Bse::SNet

+ supports_user_synths()
+ can_create_source()
+ create_source()
+ remove_source()
+ _0

Bse::SNetIface

Bse::SNetImpl

+ SNetImpl()
+ supports_user_synths()
+ auto_activate()
+ auto_activate()
+ can_create_source()
+ create_source()
+ remove_source()
```

Additional Inherited Members

Detailed Description

IDL interface class for Bse::SNet.
The documentation for this class was generated from the following file:

- bse/bseapi_interfaces.hh
2.116  Bse::SNetImpl Class Reference

Inheritance diagram for Bse::SNetImpl:

Additional Inherited Members

Detailed Description

The documentation for this class was generated from the following files:

- bse/bsesnet.hh
- bse/bsesnet.cc


### 2.117 Bse::Song Interface Reference

Interface for Track and Part objects, as well as metadata for sequencing.

```cpp
import "bseapi.idl";
```

Inheritance diagram for Bse::Song:

---

**Public Member Functions**

- **SongTiming get_timing** (int32 tick)
  
  Retrieve song timing information at a specific tick.

- **Track find_any_track_for_part** (Part part)
Find the first track that contains part, suitable to check for orphan parts.

- **Bus create_bus()**
  Create a new mixer bus for a Song.

- **void remove_bus(Bus bus)**
  Delete a mixer bus from a Song.

- **Part create_part()**
  Create a new Part in a Song.

- **void remove_part(Part part)**
  Delete a Part from a Song.

- **Track Seq list_tracks()**
  List all tracks of this song.

- **Track create_track()**
  Create a new Track for a Song.

- **void remove_track(Track track)**
  Delete a Track from a Song.

- **Bus ensure_master_bus()**
  Retrieve master output bus of a song, will create one if it doesn't exist.

- **void ensure_track_links()**
  Ensure that each part in a song is inserted into at least one track.

- **Track find_track_for_part(Part part)**
  Find a track suitable for playing notes of a given part.

- **Bus get_master_bus()**
  Retrieve master output bus of a song if it exists.

- **void synthesize_note(Track track, int32 duration, int32 note, int32 fine_tune, float64 velocity)**
  Synthesize a note on a song of an active project.

- **int64 get_shm_offset(SongTelemetry fld)**
  Offset into SharedMemory for SongTelemetry fields.

## Detailed Description
Interface for Track and Part objects, as well as meta data for sequencing.

## Member Function Documentation

### create_bus()

```
Bus Bse::Song::create_bus ( )
```
Create a new mixer bus for a Song.

### create_part()

```
Part Bse::Song::create_part ( )
```
Create a new Part in a Song.

### create_track()

```
Track Bse::Song::create_track ( )
```
Create a new Track for a Song.
ensure_master_bus()

Bus Bse::Song::ensure_master_bus ( )
Retrieve master output bus of a song, will create one if it doesn't exist.

ensure_track_links()

void Bse::Song::ensure_track_links ( )
Ensure that each part in a song is inserted into at least one track.

find_any_track_for_part()

Track Bse::Song::find_any_track_for_part ( Part part )
Find the first track that contains part, suitable to check for orphan parts.

find_track_for_part()

Track Bse::Song::find_track_for_part ( Part part )
Find a track suitable for playing notes of a given part.

get_master_bus()

Bus Bse::Song::get_master_bus ( )
Retrieve master output bus of a song if it exists.

get_shm_offset()

int64 Bse::Song::get_shm_offset ( SongTelemetry fld )
Offset into SharedMemory for SongTelemetry fields.

get_timing()

SongTiming Bse::Song::get_timing ( int32 tick )
Retrieve song timing information at a specific tick.

list_tracks()

TrackSeq Bse::Song::list_tracks ( )
List all tracks of this song.

remove_bus()

void Bse::Song::remove_bus ( Bus bus )
Delete a mixer bus from a Song.
2.118 Bse::SongIface Class Reference

IDL interface class for Bse::Song.

#include <bseapi_interfaces.hh>

remove_part()

void Bse::Song::remove_part (  
    Part part )
Delete a Part from a Song.

remove_track()

void Bse::Song::remove_track (  
    Track track )
Delete a Track from a Song.

synthesize_note()

void Bse::Song::synthesize_note (  
    Track track,  
    int32 duration,  
    int32 note,  
    int32 fine_tune,  
    float64 velocity )
Synthesize a note on a song of an active project.
The documentation for this interface was generated from the following file:

- bse/bseapi.idl
Inheritance diagram for Bse::SongIface:

Additional Inherited Members

Detailed Description

IDL interface class for Bse::Song.
The documentation for this class was generated from the following file:

- bse/bseapi_interfaces.hh
2.119  Bse::SongImpl Class Reference

Inheritance diagram for Bse::SongImpl:

Additional Inherited Members

Detailed Description

The documentation for this class was generated from the following files:

- bse/besong.hh
- bse/besong.cc
2.120  Bse::SongTiming Struct Reference

Song timing configuration.
import "bseapi.idl";

Detailed Description

Song timing configuration.
The documentation for this struct was generated from the following file:

* bse/bseapi.idl

2.121  Bse::SoundFont Interface Reference

Interface for sound fonts.
import "bseapi.idl";
Inheritance diagram for Bse::SoundFont:
Additional Inherited Members

Detailed Description

Interface for sound fonts.
The documentation for this interface was generated from the following file:

- bse/bseapi.idl

2.122 Bse::SoundFontIface Class Reference

IDL interface class for Bse::SoundFont.
#include <bseapi_interfaces.hh>

Inheritance diagram for Bse::SoundFontIface:

Additional Inherited Members

Detailed Description

IDL interface class for Bse::SoundFont.
The documentation for this class was generated from the following file:

- bse/bseapi_interfaces.hh
2.123  Bse::SoundFontImpl Class Reference

Inheritance diagram for Bse::SoundFontImpl:

![Inheritance Diagram]

Additional Inherited Members

Detailed Description

The documentation for this class was generated from the following files:

- bse/besoundfont.hh
- bse/besoundfont.cc

2.124  Bse::SoundFontRepoIface Class Reference

IDL interface class for Bse::SoundFontRepo.

#include <bseapi_interfaces.hh>
Inheritance diagram for Bse::SoundFontRepoIface:

```
Bse::SoundFontRepoIface

```
```
Bse::SoundFontRepoImpl
+ sound_fonts
+ fluid_settings
+ fluid_synth
+ SoundFontRepoImpl()
+ load_file()
+ remove_sound_font()
```

**Detailed Description**

IDL interface class for Bse::SoundFontRepo.
The documentation for this class was generated from the following file:

- bse/bseapi_interfaces.hh
2.125 Bse::SoundFontRepoImpl Class Reference

Inheritance diagram for Bse::SoundFontRepoImpl:

Additional Inherited Members

Detailed Description

The documentation for this class was generated from the following files:

- bse/besoundfontrepo.hh
- bse/besoundfontrepo.cc

2.126 Bse::Source Interface Reference

Base interface type for synthesis modules with input or output streams.

import"bseapi.idl";
Inheritance diagram for Bse::Source:

Public Member Functions

- **Source ichannel_get_osource (int32 input_channel, int32 input_joint)**
  
  Retrieve output module connected to a specific joint of an input channel.

- void **clear_inputs ()**
  
  Disconnect all module inputs.

- void **clear_outputs ()**
  
  Disconnect all module outputs.

- bool **has_output (int32 ochannel)**
  
  Check whether a module's output channel is connected.

- bool **has_outputs ()**
  
  Check whether a module has output channel connections.

- **String ichannel_blurb (int32 input_channel)**
  
  Get input channel description.

- **int32 ichannel_get_n_joints (int32 input_channel)**
  
  Retrieve the number of inputs connected to an input channel.

- **int32 ichannel_get_ochannel (int32 input_channel, int32 input_joint)**
  
  Retrieve output channel of the module connected to a specific joint of an input channel.

- **String ichannel_ident (int32 input_channel)**
  
  Get canonical input channel name.
Detailed Description

Base interface type for synthesis modules with input or output streams.

Member Function Documentation

clear_inputs()

void Bse::Source::clear_inputs()
Disconnected all module inputs.
clear_outputs()

void Bse::Source::clear_outputs ( )
Disconnect all module outputs.

create_signal_monitor()

SignalMonitor Bse::Source::create_signal_monitor ( int32 ochannel )
Create signal monitor for an output channel.

get_automation_channel()

int32 Bse::Source::get_automation_channel ( String property_name )
Get MIDI channel from an automation property.

get_automation_control()

MidiControl Bse::Source::get_automation_control ( String property_name )
Get control type from an automation property.

get_mix_freq()

int32 Bse::Source::get_mix_freq ( )
Retrieve the current mixing frequency used for probes.

has_output()

bool Bse::Source::has_output ( int32 ochannel )
Check whether a module's output channel is connected.

has_outputs()

bool Bse::Source::has_outputs ( )
Check whether a module has output channel connections.

ichannel_blurb()

String Bse::Source::ichannel_blurb ( int32 input_channel )
Get input channel description.

ichannel_get_n_joints()

int32 Bse::Source::ichannel_get_n_joints ( int32 input_channel )
Retrieve the number of inputs connected to an input channel.
ichannel_get_ochannel()

```cpp
int32 Bse::Source::ichannel_get_ochannel (  
    int32 input_channel,  
    int32 input_joint )
```
Retrieve output channel of the module connected to a specific joint of an input channel.

ichannel_get_osource()

```cpp
Source Bse::Source::ichannel_get_osource (  
    int32 input_channel,  
    int32 input_joint )
```
Retrieve output module connected to a specific joint of an input channel.

ichannel_ident()

```cpp
String Bse::Source::ichannel_ident (  
    int32 input_channel )
```
Get canonical input channel name.

ichannel_label()

```cpp
String Bse::Source::ichannel_label (  
    int32 input_channel )
```
Get input channel name.

is_joint_ichannel()

```cpp
bool Bse::Source::is_joint_ichannel (  
    String input_channel )
```
Check if an input channel is a joint (multi-connect) channel.

is_joint_ichannel_by_id()

```cpp
bool Bse::Source::is_joint_ichannel_by_id (  
    int32 input_channel )
```
Check if an input channel is a joint (multi-connect) channel.

is_prepared()

```cpp
bool Bse::Source::is_prepared ( )
```
Check whether a source is prepared for synthesis processing.

n_ichannels()

```cpp
int32 Bse::Source::n_ichannels ( )
```
Get the number of input channels of a module.

n_ochannels()

```cpp
int32 Bse::Source::n_ochannels ( )
```
Get the number of output channels of a module.
**ochannel_blurb()**

```cpp
String Bse::Source::ochannel_blurb (  
    int32 output_channel )
```

Get output channel description.

**ochannel_ident()**

```cpp
String Bse::Source::ochannel_ident (  
    int32 output_channel )
```

Get canonical output channel name.

**ochannel_label()**

```cpp
String Bse::Source::ochannel_label (  
    int32 output_channel )
```

Get output channel name.

**set_automation()**

```cpp
Error Bse::Source::set_automation (  
    String property_name,  
    int32 midi_channel,  
    MidiControl control_type )
```

Setup automation parameters for a property.

**Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>midi_channel</td>
<td>The MIDI Channel from which automation events should be received, 0 designates the default MIDI channel</td>
</tr>
<tr>
<td>control_type</td>
<td>The type of control events used for automation</td>
</tr>
</tbody>
</table>

**set_input()**

```cpp
Error Bse::Source::set_input (  
    String input_channel,  
    Source omodule,  
    String output_channel )
```

Connect a module input to another module's output.

**set_input_by_id()**

```cpp
Error Bse::Source::set_input_by_id (  
    int32 input_channel,  
    Source omodule,  
    int32 output_channel )
```

Connect a module input to another module's output.

**set_pos()**

```cpp
void Bse::Source::set_pos (  
    float64 x_pos,  
    float64 y_pos )
```
Set the x and y position of a module.
In contrast to setting the position through ordinary object property setters, this function will not update the module position if the passed in arguments are sufficiently equal to the values already set on the object. As such, it does not record an extra undo step for setting properties to values they already have and if necessary turns setting of x and y positions into an atomic undo operation.

unset_input()

Error Bse::Source::unset_input (  
    String input_channel,  
    Source omodule,  
    String output_channel )

Disconnect a module input.

unset_input_by_id()

Error Bse::Source::unset_input_by_id (  
    int32 input_channel,  
    Source omodule,  
    int32 output_channel )

Disconnect a module input.

The documentation for this interface was generated from the following file:

• bse/bseapi.idl

2.127 Bse::SourceIface Class Reference

IDL interface class for Bse::Source.
#include <bseapi_interfaces.hh>
Inheritance diagram for Bse::SourceIface:

```
Bse::Source
+ Position
  + pos_y

+ ichannel_get_osource()
+ clear_inputs()
+ clear_outputs()
+ has_output()
+ has_outputs()
+ ichannel_blurb()
+ ichannel_get_n_joints()
+ ichannel_get_ochannel()
+ ichannel_ident()
+ ichannel_label()
and 24 more...
```

```
Bse::SourceIface
```

```
Bse::SourceImpl
+ SourceImpl()
+ pos_x()
  + pos_x()
+ pos_y()
  + pos_y()
+ prepared()
+ ichannel_get_osource()
+ clear_inputs()
+ clear_outputs()
+ has_output()
and 24 more...
```

---

**Additional Inherited Members**

**Detailed Description**

IDL interface class for Bse::Source.
The documentation for this class was generated from the following file:

- bse/bseapi_interfaces.hh

Bse 0.15.0 November 2019 Beast Sound Engine
2.128  Bse::SourceImpl Class Reference

Inheritance diagram for Bse::SourceImpl:

Additional Inherited Members

Detailed Description

The documentation for this class was generated from the following files:

- bse/besource.hh
- bse/besource.cc
- bse/monitor.cc

2.129  Bse::Spinlock Class Reference

The Spinlock uses low-latency busy spinning to acquire locks.

#include <bcore.hh>
Detailed Description

The Spinlock uses low-latency busy spinning to acquire locks. This class is a thin wrapper around pthread_spin_lock() and related functions. This class supports static construction.

The documentation for this class was generated from the following file:

- bse/bcore.hh

2.130 Bse::Lib::StringFormatter Class Reference

StringFormatter - sprintf() like string formatting for C++.
#include <formatter.hh>

Static Public Member Functions

- template<LocaleContext LC = POSIX_LOCALE, class ... Args>
  static std::string format (const ArgTransform &arg_transform, const char *format, const Args &...arguments)

  Format a string according to an sprintf() format string with arguments.

Detailed Description

StringFormatter - sprintf() like string formatting for C++.
See format() for supported flags, modifiers and conversions. To find source code strings with size modifiers for possible cleanups, use: egrep ”\(\^\"([^\"]|\\")\"\)*%[0-9]*[-+0 \[[0-9]*\[hlLqjzt\]+[nSpCdiouXx←FfGgEeAa]”

Member Function Documentation

format()

template<LocaleContext LC = POSIX_LOCALE, class ... Args>
static std::string Bse::Lib::StringFormatter::format (const ArgTransform & arg_transform, const char * format, const Args & ... arguments) [inline], [static]

Format a string according to an sprintf() format string with arguments.
Refer to sprintf() for the format string details, this function is designed to serve as an sprintf() replacement and mimic its behaviour as close as possible. Supported format directive features are:

- Formatting flags (sign conversion, padding, alignment), i.e. the flags: [-#0 + ]
- Field width and precision specifications.
- Positional arguments for field width, precision and value.
- Length modifiers are tolerated: i.e. any of [hlLjztqZ].
- The conversion specifiers [spmcIdiuXxFfGgEeAa].

Additionally, arguments can be transformed after conversion by passing a std::string conversion function as arg_transform. This may e.g. be used for XML character escaping of the format argument values.

Note: Format errors, e.g. missing arguments will produce a warning on stderr and return the format string unmodified.
2.131 Bse::Strings Class Reference

Convenience Constructor for StringSeq or std::vector< std::string >
#include < strings.hh >

Inheritance diagram for Bse::Strings:

```
std::vector< std::string >
+ elements
+ assign()
+ at()
+ back()
+ begin()
+ capacity()
+ cbegin()
+ cend()
+ clear()
+ crbegin()
+ crend()
+ ... at()
+ back()
+ begin()
+ capacity()
+ cbegin()
+ cend()
+ clear()
+ crbegin()
+ crend()
and 23 more...
```

Additional Inherited Members

Detailed Description

Convenience Constructor for StringSeq or std::vector< std::string >
The documentation for this class was generated from the following files:

- bse/strings.hh
- bse/strings.cc

### 2.132 Bse::StringSeq Struct Reference

Stringeq - a variable length list of test strings.

```cpp
import"bseapi.idl";
```

**Detailed Description**

Stringeq - a variable length list of test strings.
The documentation for this struct was generated from the following file:

- bse/bseapi.idl

### 2.133 Bse::SubSynth Interface Reference

Synthesizer module for embedding (rerouting input and output) of another synthesizer network (SNet).

```cpp
import"bseapi.idl";
```
Inheritance diagram for Bse::SubSynth:

**Additional Inherited Members**

**Detailed Description**

Synthesizer module for embedding (rerouting input and output) of another synthesizer network (**SNet**). The documentation for this interface was generated from the following file:

- bse/bseapi.idl
2.134 Bse::SubSynthIface Class Reference

IDL interface class for Bse::SubSynth.
\texttt{#include <bseapi_interfaces.hh>}
Inheritance diagram for Bse::SubSynthIface:

\begin{center}
\includegraphics[width=0.5\textwidth]{inheritance_diagram.png}
\end{center}

Additional Inherited Members

Detailed Description

IDL interface class for Bse::SubSynth.
The documentation for this class was generated from the following file:

- bse/bseapi_interfaces.hh
2.135  Bse::SubSynthImpl Class Reference

Inheritance diagram for Bse::SubSynthImpl:

Additional Inherited Members

Detailed Description

The documentation for this class was generated from the following files:

- bse/besubsynth.hh
- bse/besubsynth.cc
2.136  Bse::Super Interface Reference

Base interface type for Item managers.
import "bseapi.idl";
Inheritance diagram for Bse::Super:

Additional Inherited Members

Detailed Description

Base interface type for Item managers.
The documentation for this interface was generated from the following file:

- bse/bseapi.idl

2.137  Bse::SuperIface Class Reference

IDL interface class for Bse::Super.
#include "bseapi_interfaces.hh>
Inheritance diagram for Bse::SuperIface:

```
Bse::Super
+ author
+ license

Bse::SuperIface

Bse::SuperImpl
+ SuperImpl()
+ author()
+ author()
+ license()
+ license()
```

**Additional Inherited Members**

**Detailed Description**

IDL interface class for **Bse::Super**.
The documentation for this class was generated from the following file:

- bse/bseapi_interfaces.hh
2.138  Bse::SuperImpl Class Reference

Inheritance diagram for Bse::SuperImpl:

Additional Inherited Members

Detailed Description

The documentation for this class was generated from the following files:

- bse/besuper.hh
- bse/besuper.cc

2.139  Bse::SuperSeq Struct Reference

A list of Super type objects.

Detailed Description

A list of Super type objects.

The documentation for this struct was generated from the following file:
The task registry keeps track of runtime threads for profiling and statistical purposes.

```cpp
#include <bsestartup.hh>
```

### Static Public Member Functions

- **add()**
  ```cpp
  void Bse::TaskRegistry::add (const std::string &name, int pid, int tid = -1) [static]
  ``
  Add process/thread to registry for runtime profiling.

- **remove()**
  ```cpp
  bool Bse::TaskRegistry::remove (int tid) [static]
  ``
  Remove process/thread based on thread_id.

- **update()**
  ```cpp
  void Bse::TaskRegistry::update ( ) [static]
  ``
  Issue TaskStatus.update on all tasks in registry.

- **list()**
  ```cpp
  TaskRegistry::List Bse::TaskRegistry::list ( ) [static]
  ``
  Retrieve a copy to the list of all tasks in registry.

### Detailed Description

The task registry keeps track of runtime threads for profiling and statistical purposes.

### Member Function Documentation

#### add()

```cpp
void Bse::TaskRegistry::add (const std::string & name, int pid, int tid = -1) [static]
```
Add process/thread to registry for runtime profiling.

#### list()

```cpp
TaskRegistry::List Bse::TaskRegistry::list ( ) [static]
```
Retrieve a copy to the list of all tasks in registry.

#### remove()

```cpp
bool Bse::TaskRegistry::remove (int tid) [static]
```
Remove process/thread based on thread_id.

#### update()

```cpp
void Bse::TaskRegistry::update ( ) [static]
```
Issue TaskStatus.update on all tasks in registry.

The documentation for this class was generated from the following files:

- bse/bseapi.idl
- bse/bsestartup.hh
- bse/bsestartup.cc
Acquire information about a task (process or thread) at runtime.

```cpp
#include <platform.hh>
```

### Public Member Functions

- **TaskStatus** (int pid, int tid=-1)
  
  Construct from process ID and optionally thread ID.

- **bool update()**
  
  Update status information, might return false if called too frequently.

- **String string()**
  
  Retrieve string representation of the status information.

### Public Attributes

- **int process_id**
  
  Process ID.

- **int task_id**
  
  Process ID or thread ID.

- **String name**
  
  Thread name (set by user).

- **State state**
  
  Thread state.

- **int processor**
  
  Running processor number.

- **int priority**
  
  Priority or nice value.

- **uint64 utime**
  
  Userspace time.

- **uint64 stime**
  
  System time.

- **uint64 cutime**
  
  Userspace time of dead children.

- **uint64 cstime**
  
  System time of dead children.

- **uint64 ac_stamp**
  
  Accounting stamp.

### Detailed Description

Acquire information about a task (process or thread) at runtime.

### Constructor & Destructor Documentation

**TaskStatus()**

```cpp
Bse::TaskStatus::TaskStatus (int pid, int tid = -1) [explicit]
```

Construct from process ID and optionally thread ID.
Member Function Documentation

string()

\texttt{string Bse::TaskStatus::string ( )}
Retrieve string representation of the status information.

update()

\texttt{bool Bse::TaskStatus::update ( )}
Update status information, might return false if called too frequently.

Member Data Documentation

\texttt{ac\_stamp}

\texttt{uint64 Bse::TaskStatus::ac\_stamp}
Accounting stamp.

\texttt{cstime}

\texttt{uint64 Bse::TaskStatus::cstime}
System time of dead children.

\texttt{cutime}

\texttt{uint64 Bse::TaskStatus::cutime}
Userspace time of dead children.

\texttt{name}

\texttt{String Bse::TaskStatus::name}
Thread name (set by user).

\texttt{priority}

\texttt{int Bse::TaskStatus::priority}
Priority or nice value.

\texttt{process\_id}

\texttt{int Bse::TaskStatus::process\_id}
Process ID.

\texttt{processor}

\texttt{int Bse::TaskStatus::processor}
Rrunning processor number.
2.142 Bse::Test::Timer Class Reference

```cpp
#include <testing.hh>
```

### Public Member Functions

- **Timer** (double deadline_in_secs = 0)
  
  Create a *Timer* instance, specifying an optional upper bound for test durations.

- **int64 n_reps** () const

  Number of benchmark repetitions to execute.

- **double test_elapsed** () const

  Seconds spent in benchmark()

- **double min_elapsed** () const

  Minimum time benchmarked for a callee() call.

- **double max_elapsed** () const

  Maximum time benchmarked for a callee() call.

- **template< typename Callee > double benchmark** (Callee callee)

### Detailed Description

Class for profiling benchmark tests.

**UseCase:** Benchmarking function implementations, e.g. to compare sorting implementations.

### Constructor & Destructor Documentation

state

State Bse::TaskStatus::state

Thread state.

stime

uint64 Bse::TaskStatus::stime

System time.

task_id

int Bse::TaskStatus::task_id

Process ID or thread ID.

utime

uint64 Bse::TaskStatus::utime

Userspace time.

The documentation for this struct was generated from the following files:

- bse/platform.hh
- bse/platform.cc
Timer()

Bse::Test::Timer::Timer (
    double deadline_in_secs = 0 ) [explicit]
Create a Timer() instance, specifying an optional upper bound for test durations.

Member Function Documentation

benchmark()

template<typename Callee >
double Bse::Test::Timer::benchmark ( 
    Callee callee )

Parameters

| callee | A callable function or object. Method to benchmark the execution time of callee. |

Returns

Minimum runtime in seconds,

max_elapsed()

double Bse::Test::Timer::max_elapsed ( ) const
Maximum time benchmarked for a callee() call.

min_elapsed()

double Bse::Test::Timer::min_elapsed ( ) const
Minimum time benchmarked for a callee() call.

n_reps()

int64 Bse::Test::Timer::n_reps ( ) const [inline]
Number of benchmark repetitions to execute.

test_elapsed()

double Bse::Test::Timer::test_elapsed ( ) const [inline]
Seconds spent in benchmark()
The documentation for this class was generated from the following files:

- bse/testing.hh
- bse/testing.cc

2.143 Bse::Track Interface Reference

Interface for sequencing information and links to Part objects.
import"bseapi.idl";
Inheritance diagram for Bse::Track:

```
Bse::ContextMerger

Bse::Track
+ outputs
+ get_timing()
+ insert_part()
+ remove_tick()
+ remove_link()
+ list_devices()
+ create_device()
+ device_type_info()
+ list_device_types()
+ list_parts_uniq()
+ list_parts()
+ get_part()
+ get_last_tick()
+ ensure_output()
+ 7 more...

Bse::TrackIface

Bse::TrackImpl
+ TrackImpl()
+ get_timing()
+ insert_part()
+ remove_tick()
+ remove_link()
+ list_parts_uniq()
+ list_parts()
+ get_part()
+ get_last_tick()
+ ensure_output()
+ 13 more...
```

Public Member Functions

- **SongTiming get_timing** (int32 tick)
  
  Retrieve song timing information at a specific tick.
- **int32 insert_part** (int32 tick, Part part)
  
  Insert Part into Track at tick, returns the corresponding link id.
- **void remove_tick** (int32 tick)
  
  Remove Part at specified tick from a track.
- **void remove_link** (int32 id)
Remove a specific part link by ID from a track.

- DeviceSeq list_devices()
  List devices in order of processing.
- Device create_device(String device_id)
  Create a new device with device_type.
- DeviceTypeInfo device_type_info(String device_id)
  Describe device_type.
- StringSeq list_device_types()
  List known device types.
- PartSeq list_parts_uniq()
  List all parts contained in a track.
- TrackPartSeq list_parts()
  List parts scheduled in a track, sorted by tick.
- Part get_part(int32 tick)
  Get the part starting at a specific tick position.
- int32 get_last_tick()
  Retrieve the last tick for this track.
- Error ensure_output()
  Ensure the track has an output connection to a bus.
- Source get_output_source()
  Get the output module of a track.

Public Attributes

- ItemSeq outputs
  _("Mixer busses used as output for this track.")

Detailed Description

Interface for sequencing information and links to Part objects.

Member Function Documentation

create_device()

Device Bse::Track::create_device ( String device_id )
Create a new device with device_type.

device_type_info()

DeviceTypeInfo Bse::Track::device_type_info ( String device_id )
Describe device_type.

ensure_output()

Error Bse::Track::ensure_output ()
Ensure the track has an output connection to a bus.
get_last_tick()

```cpp
int32 Bse::Track::get_last_tick ( )
```
Retrieve the last tick for this track.

get_output_source()

```cpp
Source Bse::Track::get_output_source ( )
```
Get the output module of a track.
The output of this module is the merged result from all polyphonic voices and has all track specific alterations applied.

get_part()

```cpp
Part Bse::Track::get_part ( int32 tick )
```
Get the part starting at a specific tick position.

get_timing()

```cpp
SongTiming Bse::Track::get_timing ( int32 tick )
```
Retrieve song timing information at a specific tick.

insert_part()

```cpp
int32 Bse::Track::insert_part ( int32 tick, Part part )
```
Insert Part into Track at tick, returns the corresponding link id.

list_device_types()

```cpp
StringSeq Bse::Track::list_device_types ( )
```
List known device types.

list_devices()

```cpp
DeviceSeq Bse::Track::list_devices ( )
```
List devices in order of processing.

list_parts()

```cpp
TrackPartSeq Bse::Track::list_parts ( )
```
List parts scheduled in a track, sorted by tick.

list_parts_uniq()

```cpp
PartSeq Bse::Track::list_parts_uniq ( )
```
List all parts contained in a track.
remove_link()

```cpp
void Bse::Track::remove_link (  
    int32 id  
)  
```

Remove a specific part link by ID from a track.

remove_tick()

```cpp
void Bse::Track::remove_tick (  
    int32 tick  
)  
```

Remove Part at specified tick from a track.

**Member Data Documentation**

**outputs**

`ItemSeq Bse::Track::outputs`

_("Mixer busses used as output for this track.")_

The documentation for this interface was generated from the following file:

- bse/bseapi.idl

# 2.144 Bse::TrackIface Class Reference

IDL interface class for Bse::Track.

```
#include <bseapi_interfaces.hh>
```
Inheritance diagram for Bse::TrackIface:

![Inheritance Diagram]

**Additional Inherited Members**

**Detailed Description**

IDL interface class for Bse::Track.
The documentation for this class was generated from the following file:

- `bse/bseapi_interfaces.hh`
2.145  Bse::TrackImpl Class Reference

Inheritance diagram for Bse::TrackImpl:

Additional Inherited Members

Detailed Description

The documentation for this class was generated from the following files:

- bse/bsetrack.hh
- bse/bsetrack.cc
2.146 Bse::TrackPart Struct Reference

Structure linking to a Track from within a Part.
import "bseapi.idl";

Detailed Description
Structure linking to a Track from within a Part.
The documentation for this struct was generated from the following file:
  • bse/bseapi.idl

2.147 Bse::TrackPartSeq Struct Reference

Sequence of TrackPart records.
import "bseapi.idl";

Detailed Description
Sequence of TrackPart records.
The documentation for this struct was generated from the following file:
  • bse/bseapi.idl

2.148 Bse::TrackSeq Struct Reference

Sequence of Track objects.
import "bseapi.idl";

Detailed Description
Sequence of Track objects.
The documentation for this struct was generated from the following file:
  • bse/bseapi.idl

2.149 Bse::ItemImpl::UndoDescriptor< Obj > Class Template Reference

UndoDescriptor - type safe object handle to persist undo/redo steps.
#include <bseitem.hh>

Detailed Description
template <class Obj>
class Bse::ItemImpl::UndoDescriptor< Obj >

UndoDescriptor - type safe object handle to persist undo/redo steps.
The documentation for this class was generated from the following file:
  • bse/bseitem.hh

2.150 Bse::UserMessage Struct Reference

Structure for submission of user interface messages from BSE.
import "bseapi.idl";
Public Attributes

- **UserMessageType utype**
  Severity classification for this message.
- **String title**
  Usually GUI window title.
- **String text1**
  Primary message to the user, should be limited to 80-100 chars.
- **String text2**
  Explanatory (secondary) message no limitations recommended.
- **String text3**
  Possibly (technical) details or machine error message.
- **String label**
  Message class label, used to enable/disable this type of message.

Detailed Description

Structure for submission of user interface messages from BSE.

Member Data Documentation

**label**

`String Bse::UserMessage::label`
Message class label, used to enable/disable this type of message.

**text1**

`String Bse::UserMessage::text1`
Primary message to the user, should be limited to 80-100 chars.

**text2**

`String Bse::UserMessage::text2`
Explanatory (secondary) message no limitations recommended.

**text3**

`String Bse::UserMessage::text3`
Possibly (technical) details or machine error message.

**title**

`String Bse::UserMessage::title`
Usually GUI window title.

**utype**

`UserMessageType Bse::UserMessage::utype`
Severity classification for this message.

The documentation for this struct was generated from the following file:

- `bse/bseapi.idl`
2.151 Bse::Wave Interface Reference

Interface for PCM wave samples.

import "bseapi.idl";
Inheritance diagram for Bse::Wave:

![Inheritance diagram]

Public Member Functions

- float64 chunk_get_mix_freq (int32 chunk_index)
  
  *Retrieve mixing frequency of a wave chunk.*

- float64 chunk_get_osc_freq (int32 chunk_index)
Retrieve oscillating frequency of a wave chunk.

- `int32 n_wave_chunks()`
  Get the number of wave chunks of a wave.

- `EditableSample use_editable(int32 chunk_index)`
  Retrieve an editable sample object for a wave chunk.

Detailed Description

Interface for PCM wave samples.

Member Function Documentation

chunk_get_mix_freq()

```cpp
float64 Bse::Wave::chunk_get_mix_freq (int32 chunk_index)
```
Retrieve mixing frequency of a wave chunk.

chunk_get_osc_freq()

```cpp
float64 Bse::Wave::chunk_get_osc_freq (int32 chunk_index)
```
Retrieve oscillating frequency of a wave chunk.

n_wave_chunks()

```cpp
int32 Bse::Wave::n_wave_chunks ()
```
Get the number of wave chunks of a wave.

use_editable()

```cpp
EditableSample Bse::Wave::use_editable (int32 chunk_index)
```
Retrieve an editable sample object for a wave chunk.

The documentation for this interface was generated from the following file:

- `bse/bseapi.idl`

2.152 Bse::Wavelface Class Reference

IDL interface class for Bse::Wave.
#include <bseapi_interfaces.hh>
Inheritance diagram for Bse::WaveImpl:

```
+ WaveImpl()
+ chunk_get_mix_freq()
+ chunk_get_osc_freq()
+ n_wave_chunks()
+ use_editable()
```

### Additional Inherited Members

### Detailed Description

IDL interface class for `Bse::Wave`. The documentation for this class was generated from the following file:

- `bse/bseapi_interfaces.hh`
Inheritance diagram for Bse::WaveImpl:

Additional Inherited Members

Detailed Description

The documentation for this class was generated from the following files:

- bse/bsewave.hh
- bse/bsewave.cc
2.154  Bse::WaveOsc Interface Reference

Oscillator module for wave files.
import "bseapi.idl";

Inheritance diagram for Bse::WaveOsc:

```
Bse::Source
+ Position
+ pos_y
+ ichannel_get_osource()
+ clear_inputs()
+ clear_outputs()
+ has_output()
+ has_outputs()
+ ichannel_blurb()
+ ichannel_get_n_joints()
+ ichannel_get_ochannel()
+ ichannel_ident()
+ ichannel_label()
and 18 more...

Bse::WaveOsc
+ sync_seek_perc()
+ request_pcm_position()
+ set_from_editable_sample()

Bse::WaveOscIface

Bse::WaveOscImpl
+ WaveOscImpl()
+ request_pcm_position()
+ set_from_editable_sample()
+ sync_seek_perc()
```

Public Member Functions

- void sync_seek_perc (float64 pos_perc, WaveOscSeq wosc_seq)

  Seek a list of wave oscillators to a pcm position given in percentage. The oscillators will seek to the given position synchronously.
Detailed Description

Oscillator module for wave files.

Member Function Documentation

request_pcm_position()

```cpp
void Bse::WaveOsc::request_pcm_position ( )
```

Request emission of the ::notify_pcm_position signal.

set_from_editable_sample()

```cpp
void Bse::WaveOsc::set_from_editable_sample (EditableSample esample)
```

Set wave to play from editable sample, bypassing undo and storage mechanisms.

sync_seek_perc()

```cpp
void Bse::WaveOsc::sync_seek_perc ( 
    float64 pos_perc,
    WaveOscSeq wosc_seq )
```

Seek a list of wave oscillators to a pcm position given in percentage. The oscillators will seek to the given position synchronously. The documentation for this interface was generated from the following file:

- bse/bseapi.idl

2.155 Bse::WaveOscIface Class Reference

IDL interface class for Bse::WaveOsc.

#include <bseapi_interfaces.hh>
Inheritance diagram for Bse::WaveOscIface:

Additional Inherited Members

Detailed Description

IDL interface class for Bse::WaveOsc.
The documentation for this class was generated from the following file:

- bse/bseapi_interfaces.hh
2.156 Bse::WaveOscImpl Class Reference

Inheritance diagram for Bse::WaveOscImpl:

Additional Inherited Members

Detailed Description

The documentation for this class was generated from the following files:

- bse/bsewaveosc.hh
- bse/bsewaveosc.cc

2.157 Bse::WaveOscSeq Struct Reference

A list of part note events.

import "bseapi.idl";
Detailed Description
A list of part note events.
The documentation for this struct was generated from the following file:
  • bse/bseapi.idl

2.158   Bse::WaveRepo Interface Reference

Interface serving as container for Wave objects.
import"bseapi.idl";
Inheritance diagram for Bse::WaveRepo:

Public Member Functions
  • Error load_file (String file_name)
    Load wave from file.
  • void remove_wave (Wave wave)
    Remove a wave from repository.
Detailed Description

Interface serving as container for Wave objects.

Member Function Documentation

load_file()

Error Bse::WaveRepo::load_file (
  String file_name)
Load wave from file.

remove_wave()

void Bse::WaveRepo::remove_wave ( 
  Wave wave)
Remove a wave from repository.
The documentation for this interface was generated from the following file:

- bse/bseapi.idl

2.159 Bse::WaveRepoIface Class Reference

IDL interface class for Bse::WaveRepo.
#include <bseapi_interfaces.hh>
Inheritance diagram for Bse::WaveRepoIface:

**Additional Inherited Members**

**Detailed Description**

IDL interface class for **Bse::WaveRepo**.
The documentation for this class was generated from the following file:

- bse/bseapi Interfaces.hh
2.160  Bse::WaveRepoImpl Class Reference

Inheritance diagram for Bse::WaveRepoImpl:

---

Additional Inherited Members

Detailed Description

The documentation for this class was generated from the following files:

- bse/bsewaverepo.hh
- bse/bsewaverepo.cc
Chapter 3

File Documentation

3.1  bse/bseapi.idl File Reference

Public BSE interface classes.

Classes

• struct Bse::StringSeq
  StringSeq - a variable length list of test strings.

• struct Bse::PixelSeq
  Representation of an image pixel sequence in ARGB format.

• struct Bse::Icon
  Representation of an icon pixel image.

• struct Bse::AuxData
  AuxData - record to describe entity attributes with "key = value" strings.

• struct Bse::AuxDataSeq
  AuxDataSeq - a variable length list of AuxData records.

• struct Bse::UserMessage
  Structure for submission of user interface messages from BSE.

• struct Bse::SongTiming
  Song timing configuration.

• struct Bse::NoteDescription
  A note description provides all needed details about a specific note. ".

• interface Bse::Object
  Base type for all new style C+ + objects.

• interface Bse::LegacyObject
  Base type for all legacy objects, derived from struct BseObject.

• struct Bse::ItemSeq
  A list of Item or derived objects.

• struct Bse::PropertyCandidates
  A list of items suitable to set as a specific property value.

• interface Bse::Item
  Base interface type for objects that can be added to a container.

• struct Bse::PartNote
  Part specific note event representation.

• struct Bse::PartNoteSeq
  A list of part note events.

• struct Bse::PartControl
  Part specific control event representation.

• struct Bse::PartControlSeq
A list of part control events.

- **interface Bse::Part**
  
  Data interface for containment of piano notes and MIDI effects.

- **struct Bse::PartSeq**
  
  A list of Part or derived types.

- **struct Bse::FloatSeq**
  
  A list of floating point values.

- **struct Bse::SharedMemory**
  
  Descriptor for a shared memory region.

- **struct Bse::ProbeFeatures**
  
  Bits representing a selection of probe sample data features.

- **interface Bse::SignalMonitor**
  
  Interface for monitoring output signals.

- **interface Bse::Source**
  
  Base interface type for synthesis modules with input or output streams.

- **interface Bse::Container**
  
  Base interface type for containers of Item derived types.

- **interface Bse::ContextMerger**
  
  Source module for merging multiple synthesis contexts, used to implement polyphony.

- **interface Bse::Super**
  
  Base interface type for Item managers.

- **struct Bse::SuperSeq**
  
  A list of Super type objects.

- **interface Bse::SNet**
  
  Base interface type for all kinds of synthesis networks.

- **interface Bse::CSynth**
  
  Customizable synthesis (filter) network container.

- **interface Bse::SubSynth**
  
  Synthesizer module for embedding (rerouting input and output) of another synthesizer network (SNet).

- **struct Bse::ModuleTypeInfo**
  
  Info for module types.

- **class Bse::Module**
  
  Interface for the encapsulation of audio processors.

- **struct Bse::DeviceTypeInfo**
  
  Info for device types.

- **interface Bse::Device**
  
  Interface for the encapsulation of audio processors.

- **interface Bse::Track**
  
  Interface for sequencing information and links to Part objects.

- **struct Bse::TrackSeq**
  
  Sequence of Track objects.

- **struct Bse::PartLink**
  
  Record representing the use of a Part within a Track at a specific position.

- **struct Bse::PartLinkSeq**
  
  Sequence of PartLink records.

- **struct Bse::TrackPart**
  
  Structure linking to a Track from within a Part.

- **struct Bse::TrackPartSeq**
  
  Sequence of TrackPart records.

- **interface Bse::Bus**
  
  Interface for effect stacks and per-track audio signal routing to the master output.

- **interface Bse::Song**
### Modules

- **Bse**

  The *Bse* namespace contains all functions of the synthesis engine.

### Enumerations

- **enum Bse::MonitorField**

  ```
  B::F64_GENERATION, B::F32_MIN, B::F32_MAX, B::F32_DB_SPL,
  B::F32_DB_TIP, B::END_BYTE
  ```

  Offsets for signal monitoring fields in bytes, field type and size is used as prefix.

- **enum Bse::UserMessageType**

  ```
  B::ERROR, B::WARNING, B::INFO, B::DEBUG
  ```

  Offsets for signal monitoring fields in bytes, field type and size is used as prefix.

- **enum Bse::SongTelemetry**

  ```
  B::I32_TICK_POINTER, B::BYTECOUNT
  ```

  Enumeration describing the current activation and playback state of a project.
Variables

- Const Bse::KAMMER_NOTE
  Value represents unparsable/unknown notes.
- Const Bse::KAMMER_FREQ
  Kammer note, representing the kammer frequency's MIDI note value for A' or A4.
- Const Bse::KAMMER_OCTAVE
- Const Bse::MIN_OCTAVE
  Octave number for MIDI A'.
- Const Bse::MAX_OCTAVE
  Octave of MIN_NOTE.
- Const Bse::MIN_FINE_TUNE
  Octave of MAX_NOTE.

Detailed Description

Public BSE interface classes.
Bibliography


Index

- Bse, 23
- KeccakRng
  Bse::KeccakRng, 110

abspath
  Bse::Path, 50
ac stamp
  Bse::TaskStatus, 224
activate
  Bse::Project, 149
add
  Bse::TaskRegistry, 222
aligned_alloc
  Bse, 23
aligned_free
  Bse, 23
allocate_aligned_block
  Bse, 23
application_name
  Bse, 23
application_name_init
  Bse, 24
apply_idl_property
  Bse::ItemImpl, 101
as_hex
  Bse::Xms::SerializationField, 162
as node
  Bse::Xms::SerializationField, 162
attribute
  Bse::Xms::SerializationField, 162
attributes
  Bse::AuxData, 60
auto_deactivate
  Bse::Project, 149
auto_seed
  Bse::KeccakRng, 110
  Bse::Pcg32Rng, 142
basename
  Bse::Path, 50
beastbse_cachedir_cleanup
  Bse, 24
beastbse_cachedir_create
  Bse, 24
beastbseCachedir_current
  Bse, 24
binary_lookup
  Bse, 24
binary_lookup_insertion_pos
  Bse, 24
binary_lookup_sibling
  Bse, 24
breakpoint
  Bse, 25
breakpoint
  Bse, 25
broadcast_shm.fragments
  Bse::Server, 170
new inplace, 30
posix_locale_strtold, 31
print_backtrace, 31
printer, 31
printout, 31
program alias, 31
program alias init, 31
program_cwd, 31
ProjectState, 22
random float, 32
random frange, 32
random int64, 32
random_range, 32
random nonce, 32
random secret, 32
release_aligned_block, 32
runpath, 32
sha3_224_hash, 33
sha3_256_hash, 33
sha3_384_hash, 33
sha3_512_hash, 33
shake128_hash, 33
sha256_hash, 33
shared_ptr cast, 34
SongTelemetry, 22
String, 20
string canonify, 34
string capitalize, 35
string casecmp, 35
string casefold, 35
string cmp, 35
string cmp ucid, 35
string endswith, 35
string format, 35
string from bool, 35
string from cquote, 36
string from double, 36
string from double_vector, 36
string from errno, 36
string from float, 36
string from int, 36
string from pretty_function_name, 36
string from type, 36
string from uint, 37
string from unicode, 37
string has int, 37
string hexdump, 37
string is canonified, 37
string is ucid, 37
string join, 37
string locale format, 38
string locale vprintf, 38
string listap, 38
string match identifier, 38
string match identifier tail, 38
string multiply, 38
string normalize nfc, 38
string normalize ndf, 39
string normalize nfkc, 39
string normalize nfkd, 39
string option check, 39
string option get, 39
INDEX

auto_seed, 110
bit_capacity, 110
discard, 110
forget, 110
generate, 110
KeccakRng, 110
max, 110
min, 110
n_nums, 111
operator!=, 111
operator<<, 112
operator>>, 112
operator(), 111
operator==, 112
random, 111
result_type, 109
seed, 111
xor_seed, 111

Bse::LegacyObject, 112
debug_name, 114
find_typedata, 114
proxy_id, 114
unique_id, 114

Bse::LegacyObjectIface, 114
Bse::LegacyObjectImpl, 116
Bse::Lib, 49
Bse::Lib::KeccakF1600, 104
type, 104
permute, 104
reset, 105

Bse::Lib::ScopedLocale, 159
Bse::Lib::ScopedPosixLocale, 159
posix_locale, 160

Bse::Lib::StringFormatter, 213

Bse::MidiNotifier, 116
Bse::MidiNotifierIface, 118
Bse::MidiNotifierImpl, 119
Bse::MidiSynth, 119
Bse::MidiSynthIface, 121
Bse::MidiSynthImpl, 122

Bse::Module, 122
generation, 124
module_type, 124
module_type_info, 124

Bse::Moduleface, 124
Bse::ModuleImpl, 126
Bse::ModuleTypeInfo, 127
Bse::NoteDescription, 127

Bse::Object, 127
find_prop, 128
generation, 128
list_props, 128
notify, 128
set_prop, 128

Bse::Objectface, 129
Bse::ObjectImpl, 130

Bse::Part, 130
change_control, 133
change_note, 133
check_overlap, 133
delete_event, 133
deselect_controls, 133
deselect_event, 133
deselect_notes, 133

get_channel_controls, 134
get_controls, 134
generate_last_tick, 134
generate_max_note, 134
generate_min_note, 134
get_notes, 134
get_timing, 134
insert_control, 134
insert_note, 135
insert_note_auto, 135
is_event_selected, 135
list_controls, 135
list_links, 135
list_notes, 135
list_notes_crossing, 135
list_notes_within, 135
list_selected_controls, 136
list_selected_notes, 136
queue_controls, 136
queue_notes, 136
reset, 136
select_event, 136
select_notes, 137

Bse::PartControl, 137
Bse::PartControlSeq, 137
Bse::PartFace, 138
Bse::PartImpl, 139
Bse::PartLink, 140
Bse::PartLinkSeq, 140
Bse::PartNote, 140
Bse::PartNoteSeq, 140
Bse::PartSeq, 140
Bse::Path, 49

Bse::PcmWriter, 142
Bse::PcmWriterIface, 144
Bse::PcmWriterImpl, 145

Bse::PixelSeq, 145
Bse::ProbeFeatures, 146
Bse::Procedure, 54
Bse::Project, 146
activate, 149
auto_deactivate, 149
can_play, 149
change_name, 149
clean_dirty, 149
clear_undo, 149
create_csynth, 149
create_midi_synth, 150
create_song, 150
deactivate, 150
generate_midi_notifier, 150
generate_sound_font_repo, 150
generate_state, 150
generate_supers, 150
generate_wave_repo, 150
import_midi_file, 150
inject_midi_control, 151
is_active, 151
is_dirty, 151
is_playing, 151
play, 151
redo, 151
redo_depth, 151
remove_snet, 151
restore_from_file, 151
start_playback, 152
State, 152
stop, 152
stop_playback, 152
store, 152
store_bse, 152
undo, 152
undo_depth, 152

Bse::Projectface, 153
Bse::ProjectImpl, 155
Bse::PropertyCandidates, 156
Bse::Re, 54

Bse::Resampler2, 156
delay, 157
find_precision_for_bits, 157
order, 157
precision_name, 157
process_block, 158
Resampler2, 157
reset, 158

Bse::SHA3_224, 178
digest, 179
reset, 179
SHA3_224, 178
update, 179

Bse::SHA3_256, 179
digest, 180
reset, 180
SHA3_256, 179
update, 180

Bse::SHA3_384, 180
digest, 181
reset, 181

Bse 0.15.0 November 2019 Beast Sound Engine
INDEX

get_name
  Bse::Item, 96
get_name_or_type
  Bse::Item, 96
get_notes
  Bse::Part, 134
get_ochannel
  Bse::SignalMonitor, 187
get_osce_freq
  Bse::EditableSample, 88
get_osource
  Bse::SignalMonitor, 187
get_output_source
  Bse::Track, 229
get_parent
  Bse::Item, 96
get_part
  Bse::Track, 229
group_undo
  Bse::Item, 97
get
  Bse::Xms::SerializationNode, 165
has_output
  Bse::Source, 207
gets
  Bse::Source, 207
hex
  Bse::Xms::SerializationField, 162
ichannel_blurb
  Bse::Source, 207
ichannel_get_n_joints
  Bse::Source, 207
ichannel_get_ochannel
  Bse::Source, 207
ichannel_get_osource
  Bse::Source, 208
ichannel_ident
  Bse::Source, 208
ichannel_label
  Bse::Source, 208
icon_from_pixstream
  Bse, 29
icon_sanitize
  Bse, 29
import_midi_file
  Bse::Project, 150
in
  Bse::Xms::SerializationNode, 165
in_load
  Bse::Xms::SerializationNode, 165
in_save
  Bse::Xms::SerializationNode, 165
info
  Bse, 29
init
  Bse::Test, 55
init_async
  Bse, 30
init_glue_context
  Bse, 30
init_needed
  Bse, 30
init_server_instance
  Bse, 30
inject_midi_control
  Bse::Project, 151
insert_control
  Bse::Part, 134
insert_note
  Bse::Part, 135
insert_note_auto
  Bse::Part, 135
insert_part
  Bse::Track, 229
int16
  Bse, 20
int32
  Bse, 20
int64
  Bse, 20
int8
  Bse, 20
internal
  Bse::Item, 97
is_active
  Bse::Project, 151
is_dirty
  Bse::Project, 151
is_event_selected
  Bse::Part, 135
is_joint_ichannel
  Bse::Source, 208
is_joint_ichannel_by_id
  Bse::Source, 208
is_playing
  Bse::Project, 151
is_prepared
  Bse::Source, 208
isabs
  Bse::Path, 52
isdirname
  Bse::Path, 53
KAMMER_FREQ
  Bse, 47
KAMMER_NOTE
  Bse, 47
KAMMER_OCTAVE
  Bse, 47
KecckakCryptoRng
  Bse::KecckakCryptoRng, 103
KecckakF1600
  Bse::Lib::KecckakF1600, 104
KecckakFastRng
  Bse::KecckakFastRng, 106
KecckakGoodRng
  Bse::KecckakGoodRng, 107
KecckakRng
  Bse::KecckakRng, 110
label
  Bse::UserMessage, 234
last_project
  Bse::Server, 173
list
  Bse::TaskRegistry, 222
list_children
  Bse::Container, 68
list_controls
  Bse::Part, 135
list_device_types
  Bse::Device, 81
list_devices
  Bse::Track, 229
list_links
  Bse::Part, 135
list_midi_drivers
  Bse::Server, 173
list_module_types
  Bse::Device, 81
list_modules
  Bse::Device, 81
list_notes_crossing
  Bse::Part, 135
list_notes_within
  Bse::Part, 135
list_parts

Bse 0.15.0 November 2019 Beast Sound Engine
<table>
<thead>
<tr>
<th>Bse::TaskRegistry</th>
<th>Bse::TaskStatus</th>
</tr>
</thead>
<tbody>
<tr>
<td>url_show</td>
<td>Bse::Test</td>
</tr>
<tr>
<td>use</td>
<td>utf8_to_unicode</td>
</tr>
<tr>
<td>use_editable</td>
<td>utf8len</td>
</tr>
<tr>
<td>user_home</td>
<td>utime</td>
</tr>
<tr>
<td>Bse::Wave</td>
<td>Bse::TaskStatus</td>
</tr>
<tr>
<td>Bse::Path</td>
<td>Bse::UserMessage</td>
</tr>
<tr>
<td>Bse::UserMessage</td>
<td>Bse::UserMessage</td>
</tr>
<tr>
<td>Bse::Xms::SerializationNode</td>
<td>Bse::Xms::SerializationNode</td>
</tr>
<tr>
<td>write_xml</td>
<td>xor_seed</td>
</tr>
<tr>
<td>zintern_decompress</td>
<td>zintern_free</td>
</tr>
</tbody>
</table>

**Bse::Source**
- unset_input
- unset_input_by_id

**Bse::Item**
- use
- use_editable

**Bse::Wave**
- Bse::Wave

**Bse::Path**
- user_home

**Bse::UserMessage**
- utype

**Bse::Test**
- verbose

**Bse::UserMessage**
- version

**Bse::Xms::SerializationNode**
- write_xml

**Bse::Xms::SerializationNode**
- xor_seed