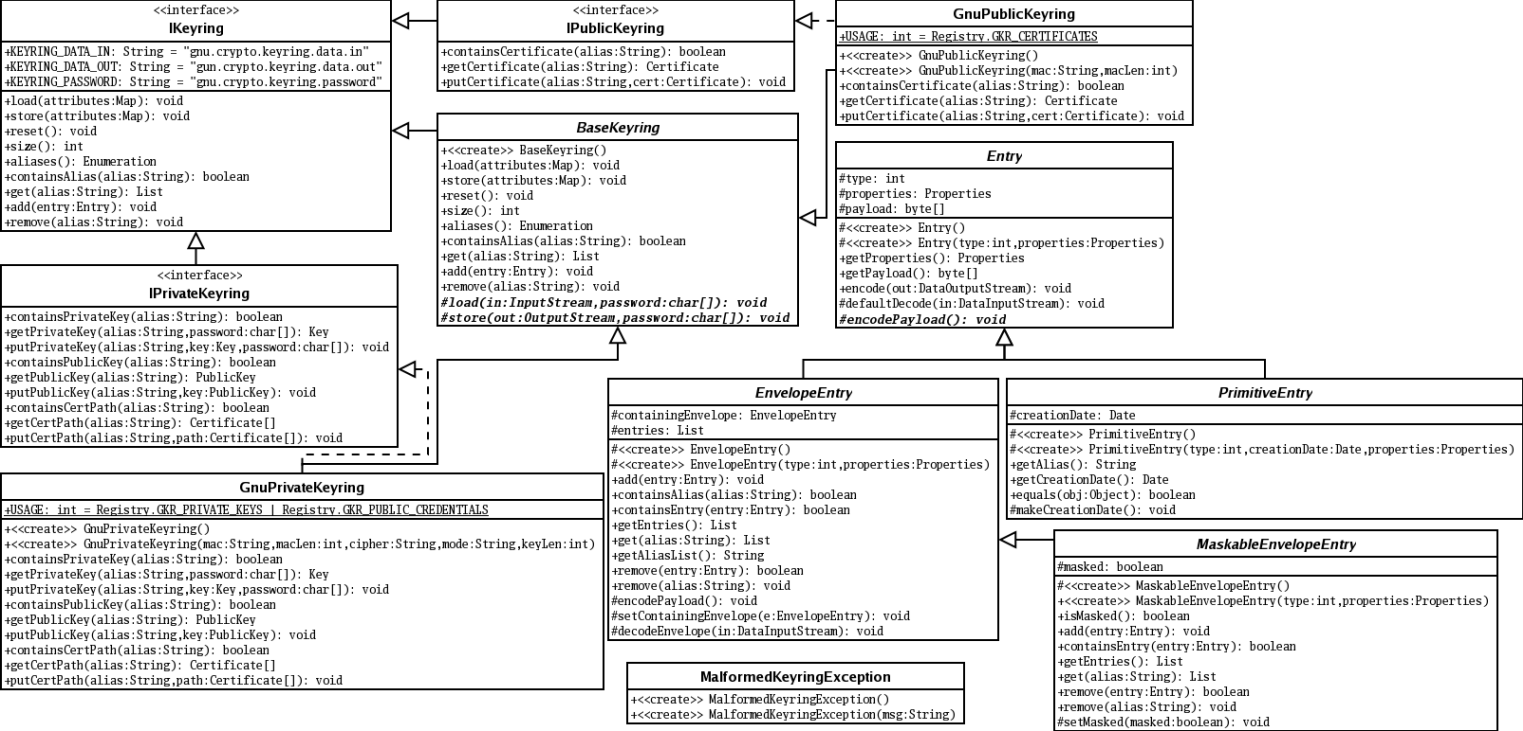
<<interface>>
IKeyring
+KEYRING_DATA_IN: String = "gnu.crypto.keyring.data.in"
+KEYRING_DATA_OUT: String = "gun.crypto.keyring.data.out"
+KEYRING_PASSWORD: String = "gnu.crypto.keyring.password"
+load(attributes:Map): void
+store(attributes:Map): void
+reset(): void
+size(): int
+aliases(): Enumeration
+containsAlias(alias:String): boolean
+get(alias:String): List
+add(entry:Entry): void
+remove(alias:String): void
<<interface>>
IPublicKeyring
+containsCertificate(alias:String): boolean
+getCertificate(alias:String): Certificate
+putCertificate(alias:String,cert:Certificate): void
GnuPublicKeyring
+USAGE: int = Registry.GKR_CERTIFICATES
+<<create>> GnuPublicKeyring()
+<<create>> GnuPublicKeyring(mac:String,macLen:int)
+containsCertificate(alias:String): boolean
+getCertificate(alias:String): Certificate
+putCertificate(alias:String,cert:Certificate): void
BaseKeyring
+<<create>> BaseKeyring()
+load(attributes:Map): void
+store(attributes:Map): void
+reset(): void
+size(): int
+aliases(): Enumeration
+containsAlias(alias:String): boolean
+get(alias:String): List
+add(entry:Entry): void
+remove(alias:String): void
#load(in:InputStream,password:char[]): void
#store(out:OutputStream,password:char[]): void
Entry
#type: int
#properties: Properties
#payload: byte[]
#<<create>> Entry()
#<<create>> Entry(type:int,properties:Properties)
+getProperties(): Properties
+getPayload(): byte[]
+encode(out:DataOutputStream): void
#defaultDecode(in:DataInputStream): void
#encodePayload(): void
<<interface>>
IPrivateKeyring
+containsPrivateKey(alias:String): boolean
+getPrivateKey(alias:String,password:char[]): Key
+putPrivateKey(alias:String,key:Key,password:char[]): void
+containsPublicKey(alias:String): boolean
+getPublicKey(alias:String): PublicKey
+putPublicKey(alias:String,key:PublicKey): void
+containsCertPath(alias:String): boolean
+getCertPath(alias:String): Certificate[]
+putCertPath(alias:String,path:Certificate[]): void
EnvelopeEntry
#containingEnvelope: EnvelopeEntry
#entries: List
#<<create>> EnvelopeEntry()
#<<create>> EnvelopeEntry(type:int,properties:Properties)
+add(entry:Entry): void
+containsAlias(alias:String): boolean
+containsEntry(entry:Entry): boolean
+getEntries(): List
+get(alias:String): List
+getAliasList(): String
+remove(entry:Entry): boolean
+remove(alias:String): void
#encodePayload(): void
#setContainingEnvelope(e:EnvelopeEntry): void
#decodeEnvelope(in:DataInputStream): void
PrimitiveEntry
#creationDate: Date
#<<create>> PrimitiveEntry()
#<<create>> PrimitiveEntry(type:int,creationDate:Date,properties:Properties)
+getAlias(): String
+getCreationDate(): Date
+equals(obj:Object): boolean
#makeCreationDate(): void
GnuPrivateKeyring
+USAGE: int = Registry.GKR_PRIVATE_KEYS | Registry.GKR_PUBLIC_CREDENTIALS
+<<create>> GnuPrivateKeyring()
+<<create>> GnuPrivateKeyring(mac:String,macLen:int,cipher:String,mode:String,keyLen:int)
+containsPrivateKey(alias:String): boolean
+getPrivateKey(alias:String,password:char[]): Key
+putPrivateKey(alias:String,key:Key,password:char[]): void
+containsPublicKey(alias:String): boolean
+getPublicKey(alias:String): PublicKey
+putPublicKey(alias:String,key:PublicKey): void
+containsCertPath(alias:String): boolean
+getCertPath(alias:String): Certificate[]
+putCertPath(alias:String,path:Certificate[]): void
MaskableEnvelopeEntry
#masked: boolean
#<<create>> MaskableEnvelopeEntry()
+<<create>> MaskableEnvelopeEntry(type:int,properties:Properties)
+isMasked(): boolean
+add(entry:Entry): void
+containsEntry(entry:Entry): boolean
+getEntries(): List
+get(alias:String): List
+remove(entry:Entry): boolean
+remove(alias:String): void
#setMasked(masked:boolean): void
MalformedKeyringException
+<<create>> MalformedKeyringException()
+<<create>> MalformedKeyringException(msg:String)