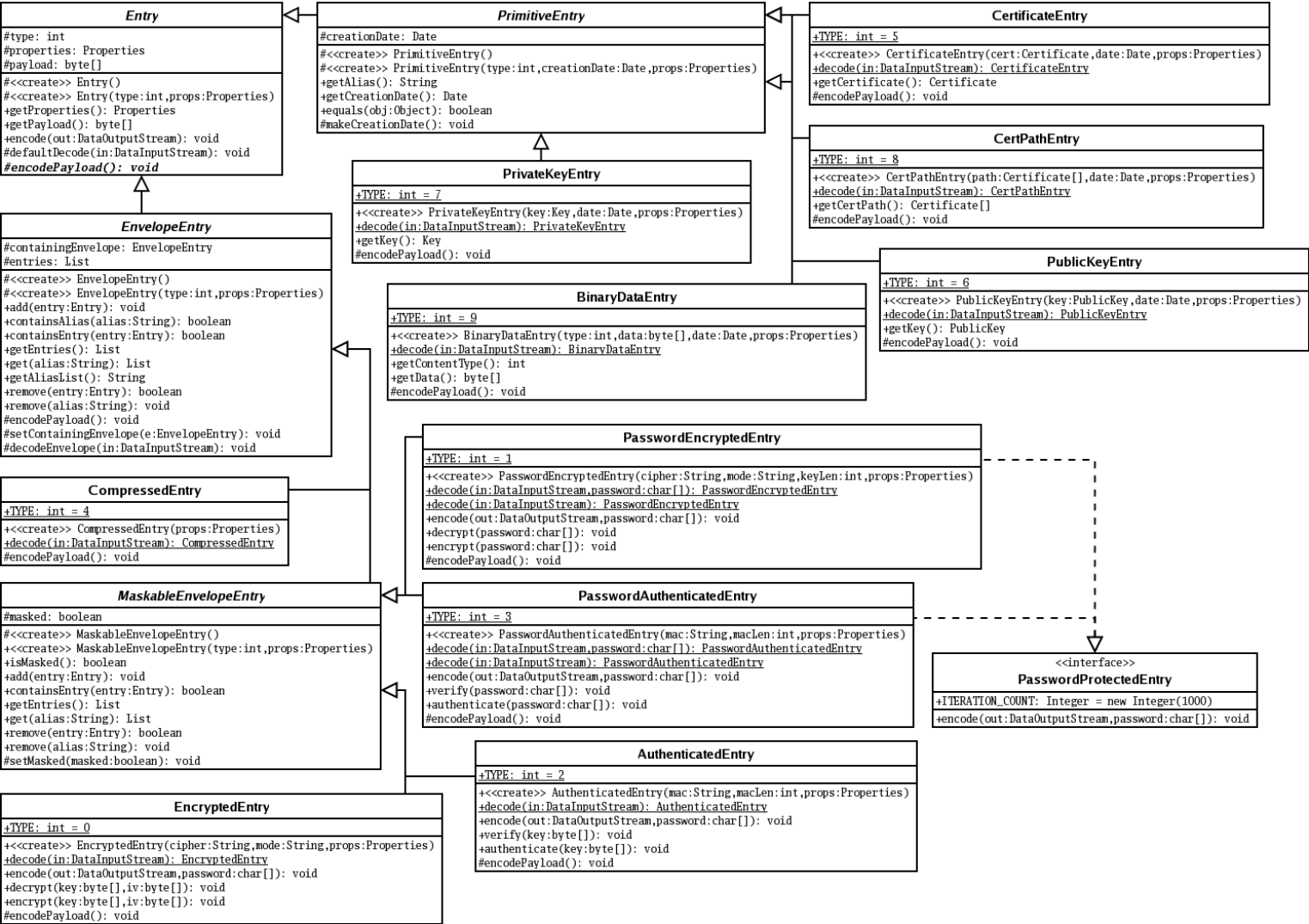
Entry
#type: int
#properties: Properties
#payload: byte[]
#<<create>> Entry()
#<<create>> Entry(type:int,props:Properties)
+getProperties(): Properties
+getPayload(): byte[]
+encode(out:DataOutputStream): void
#defaultDecode(in:DataInputStream): void
#encodePayload(): void
PrimitiveEntry
#creationDate: Date
#<<create>> PrimitiveEntry()
#<<create>> PrimitiveEntry(type:int,creationDate:Date,props:Properties)
+getAlias(): String
+getCreationDate(): Date
+equals(obj:Object): boolean
#makeCreationDate(): void
CertificateEntry
+TYPE: int = 5
+<<create>> CertificateEntry(cert:Certificate,date:Date,props:Properties)
+decode(in:DataInputStream): CertificateEntry
+getCertificate(): Certificate
#encodePayload(): void
CertPathEntry
+TYPE: int = 8
+<<create>> CertPathEntry(path:Certificate[],date:Date,props:Properties)
+decode(in:DataInputStream): CertPathEntry
+getCertPath(): Certificate[]
#encodePayload(): void
PrivateKeyEntry
+TYPE: int = 7
+<<create>> PrivateKeyEntry(key:Key,date:Date,props:Properties)
+decode(in:DataInputStream): PrivateKeyEntry
+getKey(): Key
#encodePayload(): void
PublicKeyEntry
+TYPE: int = 6
+<<create>> PublicKeyEntry(key:PublicKey,date:Date,props:Properties)
+decode(in:DataInputStream): PublicKeyEntry
+getKey(): PublicKey
#encodePayload(): void
EnvelopeEntry
#containingEnvelope: EnvelopeEntry
#entries: List
#<<create>> EnvelopeEntry()
#<<create>> EnvelopeEntry(type:int,props: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
BinaryDataEntry
+TYPE: int = 9
+<<create>> BinaryDataEntry(type:int,data:byte[],date:Date,props:Properties)
+decode(in:DataInputStream): BinaryDataEntry
+getContentType(): int
+getData(): byte[]
#encodePayload(): void
PasswordEncryptedEntry
+TYPE: int = 1
+<<create>> PasswordEncryptedEntry(cipher:String,mode:String,keyLen:int,props:Properties)
+decode(in:DataInputStream,password:char[]): PasswordEncryptedEntry
+decode(in:DataInputStream): PasswordEncryptedEntry
+encode(out:DataOutputStream,password:char[]): void
+decrypt(password:char[]): void
+encrypt(password:char[]): void
#encodePayload(): void
CompressedEntry
+TYPE: int = 4
+<<create>> CompressedEntry(props:Properties)
+decode(in:DataInputStream): CompressedEntry
#encodePayload(): void
MaskableEnvelopeEntry
#masked: boolean
#<<create>> MaskableEnvelopeEntry()
+<<create>> MaskableEnvelopeEntry(type:int,props: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
PasswordAuthenticatedEntry
+TYPE: int = 3
+<<create>> PasswordAuthenticatedEntry(mac:String,macLen:int,props:Properties)
+decode(in:DataInputStream,password:char[]): PasswordAuthenticatedEntry
+decode(in:DataInputStream): PasswordAuthenticatedEntry
+encode(out:DataOutputStream,password:char[]): void
+verify(password:char[]): void
+authenticate(password:char[]): void
#encodePayload(): void
<<interface>>
PasswordProtectedEntry
+ITERATION_COUNT: Integer = new Integer(1000)
+encode(out:DataOutputStream,password:char[]): void
AuthenticatedEntry
+TYPE: int = 2
+<<create>> AuthenticatedEntry(mac:String,macLen:int,props:Properties)
+decode(in:DataInputStream): AuthenticatedEntry
+encode(out:DataOutputStream,password:char[]): void
+verify(key:byte[]): void
+authenticate(key:byte[]): void
#encodePayload(): void
EncryptedEntry
+TYPE: int = 0
+<<create>> EncryptedEntry(cipher:String,mode:String,props:Properties)
+decode(in:DataInputStream): EncryptedEntry
+encode(out:DataOutputStream,password:char[]): void
+decrypt(key:byte[],iv:byte[]): void
+encrypt(key:byte[],iv:byte[]): void
#encodePayload(): void