Universal Keystore Database

```
-- SQL Script for MySQL 5.0
--
-- Tables for the web-based phone emulator.
-- These tables should be fairly applicable to a native implementation as well
-- with *.UserID as the notable exception assuming there is just a single user.
--
-- Note: USERS table is defined in keycenter.sql.
--

/*=============================================*/
/*               USERKEYS Table                */
/*=============================================*/

CREATE TABLE USERKEYS
  (
    KeyID         INT             NOT NULL  AUTO_INCREMENT,            -- Each key gets a unique ID in the database
    Created       TIMESTAMP       NOT NULL  DEFAULT CURRENT_TIMESTAMP,  -- Nice to know when created
    UserID        INT             NOT NULL,                             -- Owner device of key
    Exportable    BOOLEAN         NOT NULL  DEFAULT 0,                  -- The key may exported
    Archived      BOOLEAN         NOT NULL  DEFAULT 0,                  -- The issuer has a copy of the private key
    FriendlyName  VARCHAR (50)    NULL,                                 -- Optional human-oriented ID
--
--        Only defined for PIN-protected keys
--
    PINPolicyID   INT             NULL,                                 -- Unique ID of associated PIN policy
    PINValue      BLOB            NULL,                                 -- Encrypted PIN.  NULL => Not yet defined
    PINSettable   BOOLEAN         NOT NULL  DEFAULT 1,                  -- The PIN can be set by the user?
    PINTryCount   SMALLINT        NULL,                                 -- Decremented for each error, locking at 0
--
--            User-key cryptographic data
--
    CertPath      BLOB            NULL,                                 -- Certificate path.  NULL => Not yet defined
    PrivateKey    BLOB            NULL,                                 -- Encrypted private key (sym. key = NULL)
    SecretKey     BLOB            NULL,                                 -- [Encrypted "piggybacked" symmetric key]
    SuppAlgs      TEXT            NULL,                                 -- [ -"- ] NULL => Unrestricted usage
--
    FOREIGN KEY (UserID) REFERENCES USERS (UserID),
    FOREIGN KEY (PINPolicyID) REFERENCES PINPOLICIES (PINPolicyID),
    PRIMARY KEY (KeyID)
  );


/*=============================================*/
/*              DEVICEDATA Table               */
/*=============================================*/

CREATE TABLE DEVICEDATA
  (
    UserID        INT             NOT NULL,                             -- Owner device of key
    Created       TIMESTAMP       NOT NULL  DEFAULT CURRENT_TIMESTAMP,  -- Nice to know when created
--
--            Device certificate
--
    CertPath      BLOB            NOT NULL,                             -- Certificate path
    PrivateKey    BLOB            NOT NULL,                             -- Matching encrypted private key
--
--                System PUK
--
    PUKPolicyID   INT             NOT NULL,                             -- Unique ID of PUK policy
--
    FOREIGN KEY (PUKPolicyID) REFERENCES PUKPOLICIES (PUKPolicyID),
    FOREIGN KEY (UserID) REFERENCES USERS (UserID)
  );
```

```
/*=============================================*/
/*              PUKPOLICIES Table              */
/*=============================================*/

CREATE TABLE PUKPOLICIES
  (
    PUKPolicyID   INT            NOT NULL  AUTO_INCREMENT,                -- Unique ID of PUK policy
    Created       TIMESTAMP      NOT NULL  DEFAULT CURRENT_TIMESTAMP,     -- Nice to know when created
    RetryLimit    SMALLINT       NOT NULL,                                -- PUK tries before locking the key(s)
    PUKTryCount   SMALLINT       NOT NULL,                                -- Decremented for each error, locking at 0
    Format        SMALLINT       NOT NULL,                                -- Ordinal (0..n) of "PassphraseFormats"
    PUKValue      BLOB           NULL      DEFAULT NULL,                  -- Encrypted PUK.  NULL => Not yet defined
    PRIMARY KEY (PUKPolicyID)
  );


/*=============================================*/
/*              PINPOLICIES Table              */
/*=============================================*/

CREATE TABLE PINPOLICIES
  (
    PINPolicyID   INT            NOT NULL  AUTO_INCREMENT,                -- Unique ID of PIN policy
    Created       TIMESTAMP      NOT NULL  DEFAULT CURRENT_TIMESTAMP,     -- Nice to know when created
    RetryLimit    SMALLINT       NOT NULL,                                -- PIN tries before locking the key(s)
    PUKPolicyID   INT            NOT NULL,                                -- For every PIN there is a governing PUK
--
--          User PIN set constraints
--
    Format        SMALLINT       NOT NULL,                                -- Ordinal (0..n) of "PassphraseFormats"
    MinLength     SMALLINT       NOT NULL,                                -- Shortest acceptable PIN
    MaxLength     SMALLINT       NOT NULL,                                -- Longest acceptable PIN
    Grouping      SMALLINT       NOT NULL,                                -- Ordinal (0..n) of "PINGrouping"
    PatternRestr  BLOB (32)      NULL,                                    -- "PatternRestrictions" [len + ordinals]
--
--              API control
--
    InputMeth     SMALLINT       NOT NULL,                                -- Ordinal (0..n) of "InputMethods"
    CachingSupp   BOOLEAN        NOT NULL,                                -- Caching PIN support option
--
    FOREIGN KEY (PUKPolicyID) REFERENCES PUKPOLICIES (PUKPolicyID),
    PRIMARY KEY (PINPolicyID)
  );


/*=============================================*/
/*             TYPEREGISTRY Table              */
/*=============================================*/

CREATE TABLE TYPEREGISTRY
  (
    TypeID        INT            NOT NULL  AUTO_INCREMENT,                -- Each URI gets a unique ID for references
    TypeURI       VARCHAR (256)  NOT NULL,                                -- Type = URI
--
    PRIMARY KEY (TypeID)
  );


/*=============================================*/
/*         PROPERTYBAGCONSUMERS Table          */
/*=============================================*/

CREATE TABLE PROPERTYBAGCONSUMERS
  (
--
--   There may be more than one application that wants a specific
--   property bag type, each requring a "subscription" in this table
--
    TypeID        INT            NOT NULL,                                -- Reference to type (URI) of property bag
    ImplClass     VARCHAR (256)  NOT NULL,                                -- Java implementation class of consumer
--
    FOREIGN KEY (TypeID) REFERENCES TYPEREGISTRY (TypeID)
  );
```

```
/*=============================================*/
/*              PROPERTYBAGS Table             */
/*=============================================*/

CREATE TABLE PROPERTYBAGS
  (
    PropBagID     INT            NOT NULL  AUTO_INCREMENT,                 -- Each bag instance gets a unique ID
    KeyID         INT            NOT NULL,                                 -- Owning key
    TypeID        INT            NOT NULL,                                 -- Reference to type (URI) of property bag
--
    FOREIGN KEY (KeyID) REFERENCES USERKEYS (KeyID) ON DELETE CASCADE,
    FOREIGN KEY (TypeID) REFERENCES TYPEREGISTRY (TypeID),
    PRIMARY KEY (PropBagID)
  );


/*=============================================*/
/*               PROPERTIES Table              */
/*=============================================*/

CREATE TABLE PROPERTIES
  (
    PropBagID     INT            NOT NULL,                                 -- Owning bag instance
    PropName      VARCHAR (256)  NOT NULL,                                 -- Name of the property
    PropValue     TEXT           NOT NULL,                                 -- Matching value
    Writable      BOOLEAN        NOT NULL,                                 -- True if writable
--
    FOREIGN KEY (PropBagID) REFERENCES PROPERTYBAGS (PropBagID) ON DELETE CASCADE
  );


/*=============================================*/
/*           EXTENSIONCONSUMERS Table          */
/*=============================================*/

CREATE TABLE EXTENSIONCONSUMERS
  (
--
--   There may be more than one application that wants a specific
--   extension type, each requring a "subscription" in this table.
--
    TypeID        INT            NOT NULL,                                 -- Reference to type (URI) of extension
    ImplClass     VARCHAR (256)  NOT NULL,                                 -- Java implementation class of consumer
--
    FOREIGN KEY (TypeID) REFERENCES TYPEREGISTRY (TypeID)
  );


/*=============================================*/
/*               EXTENSIONS Table              */
/*=============================================*/

CREATE TABLE EXTENSIONS
  (
    KeyID         INT            NOT NULL,                                 -- Owning key
    TypeID        INT            NOT NULL,                                 -- Each extension has a specific type
    ExtnData      BLOB           NOT NULL,                                 -- The extracted binary data
--
    FOREIGN KEY (KeyID) REFERENCES USERKEYS (KeyID) ON DELETE CASCADE,
    FOREIGN KEY (TypeID) REFERENCES TYPEREGISTRY (TypeID)
  );


/*=============================================*/
/*                LOGOTYPES Table              */
/*=============================================*/

CREATE TABLE LOGOTYPES
  (
    KeyID         INT            NOT NULL,                                 -- Owning key
    TypeID        INT            NOT NULL,                                 -- Each image has a specific type (=usage)
    ImageData     BLOB           NOT NULL,                                 -- The binary image data
    MimeType      VARCHAR (100)  NOT NULL,                                 -- The MIME type for the image
--
    FOREIGN KEY (KeyID) REFERENCES USERKEYS (KeyID) ON DELETE CASCADE,
    FOREIGN KEY (TypeID) REFERENCES TYPEREGISTRY (TypeID)
  );
```

```
/*=============================================*/
/*             AUTOSELECTIONS Table            */
/*=============================================*/

CREATE TABLE AUTOSELECTIONS
  (
    KeyID         INT            NOT NULL,                             -- Owning key
    TypeID        INT            NOT NULL,                             -- Associated application
    HostName      VARCHAR (256)  NOT NULL,                             -- The pre-selected host
--
    FOREIGN KEY (KeyID) REFERENCES USERKEYS (KeyID) ON DELETE CASCADE,
    FOREIGN KEY (TypeID) REFERENCES TYPEREGISTRY (TypeID)
  );


/*=============================================*/
/*              PROVISIONINGS Table            */
/*=============================================*/

CREATE TABLE PROVISIONINGS
  (
    ProvisionID   INT            NOT NULL  AUTO_INCREMENT,             -- Each provisioning gets a unique ID
    UserID        INT            NOT NULL,                             -- Owner of this particular provisioning
    Created       TIMESTAMP      NOT NULL  DEFAULT CURRENT_TIMESTAMP,  -- Nice to know when created
    ClientSession VARCHAR (256)  NOT NULL,                             -- The ID of the client session
    ServerSession VARCHAR (256)  NOT NULL,                             -- The ID of the server session
    SavedRequest  BLOB           NULL,                                 -- Serialized KeyOperationRequestDecoder
    DelayedDeploy INT            NULL,                                 -- Defined => Max days to wait
--
    PRIMARY KEY (ProvisionID),
    FOREIGN KEY (UserID) REFERENCES USERS (UserID) ON DELETE CASCADE
  );


/*=============================================*/
/*             PROVISIONEDKEYS Table           */
/*=============================================*/

CREATE TABLE PROVISIONEDKEYS
  (
    ProvisionID   INT            NOT NULL,                             -- Owning provisioning session
    KeyID         INT            NOT NULL,                             -- Local KeyID of provisioned key
    KeyUsage      INT            NOT NULL,                             -- Ordinal (0..n) of "KeyGen2KeyUsage"
    PublicKey     BLOB           NOT NULL,                             -- The generated public key serialized
    ServerKeyID   VARCHAR (256)  NOT NULL,                             -- The server's symbolic name
    ReplaceKeyID  INT            NULL,                                 -- Defined => Original KeyID (for update)
--
    FOREIGN KEY (ProvisionID) REFERENCES PROVISIONINGS (ProvisionID) ON DELETE CASCADE
  );


/*=============================================*/
/*               DELETEDKEYS Table             */
/*=============================================*/

CREATE TABLE DELETEDKEYS
  (
    ProvisionID   INT            NOT NULL,                             -- Owning provisioning session
    KeyID         INT            NOT NULL,                             -- KeyID of key to be deleted
--
    FOREIGN KEY (ProvisionID) REFERENCES PROVISIONINGS (ProvisionID) ON DELETE CASCADE
  );
```