Marketing Use Cases - Manage Key Lifecycle

muc601 :: View the data encryption keys stored on an LKM

PURPOSE: Enables a user to find an data encryption key stored on an LKM and view information about it. The user can view a key based on the dataset it was used to encrypt.

INITIAL STATE

- The LKM is *ready*.
- There may be saved configurations from previous sessions.
- Datasets have been encrypted.

FLOW

The User...	The System...
Selects an LKM (or LKMs) or selects a dataset.	Displays the LKM or dataset as selected.
Chooses the `view data encryption keys` command.	Displays the `view data encryption keys UI`.
	Displays all the data encryption keys the LKM holds.

Can view the following info about a data encryption key:

- Key
- ID
- Dataset
- Class
- Description
- Name
- Version
- Sequence
- Format
- Format version
- Flags
- Type
- State (includes if a key has been corrupted

• KeyObject signature value • KeyObject signature VK ID • KeyObject signature VK class • Signature • Creation time • Expiration time • Key sharing group • Owner IP address • Signed • Storage type • Algorithm version • DB flags • Domain key ID • Path • Server name • Where the key originated	
Can filter the information by one or more items.	Removes keys that don't match the filer.
Can sort the information by an item in ascending or descending order.	Displays the information in the order determined by the sort order.
Can rearrange the information in the UI.	Displays the information as arranged by the user.
Can show or hide elements of info independently.	Displays the information selected by the user.
Can give a name and save the current: • Elements • Arrangement • Filters • Sort order	Saves the configuration.
Can choose a configuration by name.	Displays the information in the configuration the user chooses.
Can delete a saved configuration.	
Clicks `ok`.	Closes the `view data encryption keys UI`.

END STATE

- Any save configurations are retained.

ALERT or ERROR STATES *(Requires a unique UI state or a message.)*

- No keys match the filter

ISSUES

- The current flow is based on the SMAI pattern for displaying metadata about an object in the system. It doesn't really work for large groups of data, like keys. We're leaving this use case as is, but we need to figure out how to create a UI that enables users to easily find, view, and act on the element they're interested in.

muc602 :: View the parent keys stored on an LKM

PURPOSE: Enables a user to find a parent key stored on an LKM and view information about it. The parent key can be either the LKM's proper parent key or an OpenKey client's parent key backed up on the LKM. The user can view parent keys on more then one LKM at a time.

INITIAL STATE

- The LKM is *ready*.
- There may be saved configurations from previous sessions.

FLOW

The User...	The System...
Selects the LKM (or LKMs) and chooses the `view parent key` command.	Displays the `view parent keys UI`. Displays all the parent keys the LKM holds.
Can view the following info about a parent key: • Key • Domain key ID • Policy key ID • Timestamp • Flags • Record ID • Record flags • Generation • Which LKM the key is on	
- - - These behaviors are the same as those in muc601 - - -	
Can filter the information by one or more items.	Removes keys that don't match the filer.
Can sort the information by an item in ascending or descending order.	Displays the information in the order determined by the sort order.
Can rearrange the information in the UI.	Displays the information as arranged by the user.
Can show or hide elements of info independently.	Displays the information selected by the user.
Can give a name and save the current: • Elements • Arrangement • Filters • Sort order	Saves the configuration.
Can choose a configuration by name. Can delete a saved configuration.	Displays the information in the configuration the user chooses.
- - -	
Clicks `ok`.	Closes the `view parent keys UI`.

END STATE

- Any saved configurations are retained.

ALERT or ERROR STATES *(Requires a unique UI state or a message.)*

- No keys match a filter.

ISSUES

- Should the system automatically check to see if any key being displayed has been corrupted? E.g., an additional attribute could be whether the key is still usable. -- Apparently, it can't since there's no `state` attribute.
- This use case may not be pertinent and may be deleted. Waiting for feedback.

muc603 :: ~~Create a data encryption key for an OpenKey client (no user action)~~ DELETED

PURPOSE: The LKM to generates data encryption keys for an OpenKey client so that data can be encrypted. An LKM creates keys automatically without any action on the part of the user.

INITIAL STATE

- The LKM is *ready*.
- The OpenKey client is ready (details TBD, but equivalent to the LKM's ready state).

FLOW

The User...	The System...
N/A	Receives a request for a data encryption key from the OpenKey client.
	Creates a key.
	Stores the key.
	Sends a copy of the key to the client.

END STATE

- The LKM has has created and stored one or more data encryption keys for the OpenKey client.
- The LKM has sent the key (or keys) to the client.

ALERT or ERROR STATES *(Requires a unique UI state or a message.)*

- TBD

ISSUES

- None

muc604 :: ~~Store keys in the LKM's database (no user action)~~ DELETED

PURPOSE: The LKM stores the data encryption keys and parent keys (if any) to keep them secure. The LKM stores keys automatically without the need for any action on the user's part.

INITIAL STATE

- The LKM is *ready*.
- The LKM has created data encryption keys.
- The OpenKey client may have parent keys.

FLOW

The User...	The System...
N/A	To come...

END STATE

- The LKM has stored the keys securely.

ALERT or ERROR STATES *(Requires a unique UI state or a message.)*

- TBD

ISSUES

- None

muc605 :: ~~Back up keys to another LKM (no user action)~~ DELETED

PURPOSE: An LKM duplicates the data encryption keys, the parent keys, and the configuration databases it has stored and sends the duplicates to another LKM for storage. The redundant set of keys and configuration databases protects them in case the LKM fails.

INITIAL STATE

- Both LKMs are *ready*.
- The two LKMs are linked.

FLOW

The User...	The System...
N/A	To come...

END STATE

- The second LKM has duplicate copies of the data encryption keys, parent keys, and configuration databased stored in its database.

ALERT or ERROR STATES *(Requires a unique UI state or a message.)*

- TBD

ISSUES

- None

muc606 :: Export data encryption keys to a file

PURPOSE: Enables a user to create a file containing backup copies of data encryption keys stored in an LKM so the user can store the file in secure storage in case the LKM fails and the keys it contains are lost. Backing up keys by exporting them to a file is necessary primarily:

- If an LKM isn't linked to another LKM
- If all the LKMs are in one physical location
- As a backup to two LKMs linked to each other

INITIAL STATE

- The LKM is *ready*.
- The LKM contains data encryption keys that haven't been backed up.

FLOW

The User...	The System...
Selects the LKM.	Limits the select to one LKM.
Chooses the **`export`** command.	Displays the **`export UI`**.
Chooses an option:	
• Export all the data encryption keys.	Shows the number of keys that will be exported.
• Export only the data encryption keys that haven't yet been exported.	Shows the number of keys that will be exported and the total number of keys.
	Checks to see if any of the keys has been corrupted.
Clicks the **`export`** button.	Displays the **`save dialog box`**.
	Provides a default file name.
Can change the name of the file.	Shows the new file name.
Chooses the location where she wants to save the file.	Displays the location.
Clicks the **`save`** button.	Saves the keys in the location the user has chosen in a signed file.
	Closes the **`export UI`**.

END STATE

- The data encryption keys stored in the LKM are saved in a signed file. (The file can be imported only into the LKM from which it was saved.)
- The LKM flags the keys that have been exported.

ALERT or ERROR STATES *(Requires a unique UI state or a message.)*

- The file can't be created.
- One or more keys have been corrupted.

ISSUES

- Archiving an OpenKey client's parent keys isn't mentioned in the PRD. Is it something Dolphin will do?

muc607 :: Import data encryption keys from a file

PURPOSE: Enables a user to use a file to restore data encryption keys to the LKM that originally created them and from which they were exported. Also enables the user to share keys that are on one LKM with another LKM when the two have a trust relationship, but aren't linked, and can't communicate over a network. The user can import all the keys or just selected keys (for example, if he's replacing a corrupted key).

INITIAL STATE

- The LKM is *ready*.
- Data encryption keys have been exported to a file.
- The user has access to the file that contains the keys.

FLOW

The User...	The System...
Selects the LKM he want to restore the data encryption keys to.	Shows the LKM he's selected.
Chooses the **`import`** command.	Displays the **`import UI`**.
Chooses the file.	
Selects either of these options: • Import all keys • Import seleted keys	
Clicks the **`open`** button.	Verifies that the selected LKM created the signed file.
If the user wants to import all the keys:	
	Imports the keys in the file.
	Checks to see if any of the keys has been corrupted.
	Tells the user when the import is complete and successful.
If the user wants to import only selected keys:	Displays the **`view keys UI`**.
Can filter the information by one or more items.	Removes keys that don't match the filer.
Can sort the information by an item in ascending or descending order.	Displays the information in the order determined by the sort order.
Can rearrange the information in the UI.	Displays the information as arranged by the user.
Can show or hide elements of info independently.	Displays the information selected by the user.
Selects the keys he wants to import and clicks **`import`**.	Imports the keys.
	Checks to see if any of the keys has been corrupted.
	Tells the user when the import is complete and successful.
Acknowledges the message.	Removes the **`message`**.
	Closes the **`import UI`**.

END STATE

- The data encryption keys are stored in the LKM.

ALERT or ERROR STATES *(Requires a unique UI state or a message.)*

- The LKM didn't create the signed file.
- The file can't be open.
- The keys in the file can't be imported.
- One or more keys have been corrupted.

ISSUES

- None

muc608 :: Delete keys

PURPOSE: Enables a user to delete data encryption keys stored on an LKM. A user deletes a key for one of two reasons:

- If a key has been compromised, the user first decrypts the data that the key has been used to decrypt, rekeys the data, then deletes the key.
- If data needs to be destroyed, deleting the key used to encrypt it effectively destroys the data.

INITIAL STATE

- The LKM is *ready*.
- The LKM has keys stored on it.

FLOW

The User...	The System...
Selects the LKM (or LKMs) that contain the keys she wants to delete.	Shows the LKM that's selected.
Chooses the **`delete keys`** command.	Displays the **`delete keys UI`**.
Finds the data encryption keys she wants to delete (see muc601).	Shows the keys the user has found.
Selects the keys she wants to delete.	Shows the keys the user has selected.
Clicks the **`delete`** button.	Displays a **`confirmation UI`**.
Confirms she wants to delete the keys.	Deletes the keys.
	Closes the **`confirmation UI`**.
	Displays a **`list`** of backup files that contain the deleted keys.
Selects a backup file from the list.	Opens the **`folder`** where the file was saved.
Can navigate to a different folder if the file was moved.	
Clicks the `delete file` button.	Deletes the file.
	Removes the **`folder UI`**.
Can delete another file.	
Clicks the **`close`** button.	Removes the **`delete keys UI`**.

END STATE

- The keys that the user has selected have been deleted from the system.
- Backup files that the user selected have been deleted from the system.
- Backup files that haven't been selected are blacklisted.

ALERT or ERROR STATES *(Requires a unique UI state or a message.)*

- The user doesn't confirm the action correctly (i.e., types in the wrong confirmation code).

ISSUES

- To show the list of the backup files that contain the deleted keys requires tracking all the backups.
- The UI suggested above doesn't scale. Perhaps a better approach is to have Dolphin look in the folder where the file was saved, and if it's there delete it. Then, after than, display a list of the backup files it can't find. There could also be an option to confirm each file before it's deleted.

muc609 :: Retire a key

PURPOSE: Enables a user to set the system so that it stops using a key to encrypt data. The key isn't delete because it's still needed to decrypt data. The user can set the system to stop using the key immediately, or to stop using it as some time in the future.

INITIAL STATE

- The LKM is *ready*.
- The system is using the key to encrypt data.

FLOW

The User...	The System...
Selects the LKM (or LKMs) that contain the key (or keys) he wants to retire.	Shows the LKM that's selected.
Chooses the **retire key** command.	Displays the **retire key UI**.
Finds the data encryption key he wants to retire (see muc601).	Shows the key the user has found.
	Checks to make sure the key hasn't been corrupted.
Either: • Chooses **retire now**. • Sets the date on which he wants the system to stop using the key to encrypt data.	
Clicks **ok**.	Executes the users choice: • Stops using the key immediately. • Records the date it will stop using the key.

END STATE

- If the user has chosen **retire now**, the system stops using the key to encrypt data, but keeps it stored so it's available to the OpenKey client to use to decrypt data.
- If the user has set a retirement date, the system keeps using the key, but flags the date when the key will retire.

ALERT or ERROR STATES *(Requires a unique UI state or a message.)*

- The key has been corrupted.

ISSUES *None

muc610 :: Rotate keys

PURPOSE: Enables a user to replace a key after it's been used for a certain period with a different key. Rotating keys enhances security by limiting the amount of data encrypted with any one key. The LKM manages rotation by keeping track of when the a key is to be retired. When the time comes, and the OpenKey client asks for a key, the LKM provides it with a new key.

INITIAL STATE

- The LKM is *ready*.

FLOW

The User...	The System...
Selects the dataset (or datasets) that's going to be encrypted with the rotated keys.	Shows the dataset as selected.
Chooses the **rotate keys** command.	Displays the **rotate keys UI**.
Sets the length of time each key is used before it's retired.	
Clicks the **ok** button.	Records the settings. Closes the **rotate keys UI**.

END STATE

- When a key reaches the end of the period it's to be used to encrypt data, the LKM replaces it with another key.

ALERT or ERROR STATES *(Requires a unique UI state or a message.)*

- None

ISSUES

- None.

muc611 :: Change a key's **state** attribute

PURPOSE: Enables a user to change certain values of a key's **state** attribute. The user can change the state of more than one key on more than one LKM at a time -- as long as all the keys are being changed to the same state. The user can change these values:

- Compromised / Secure
- ...

INITIAL STATE

- The LKM is *ready*.
- The key's `state` attribute is set.

FLOW

The User...	The System...
Selects the LKM (or LKMs) where the key (or keys) are located.	Shows the LKM as selected.
Chooses the **`change key state`** command.	Displays the **`change key state UI`**.
Finds the data encryption key she wants to retire (see muc601).	Shows the key the user has found.
	Checks to make sure the key hasn't been corrupted.
Selects the state she wants to change the key to.	
Clicks the **`change`** button.	Warns that the change can't be undone and displays a **`confirmation dialog box`**.
Confirms that she wants to make the change.	Removes the **`confirmation dialog box`**.
	Changes the state of the key.
	Shows the new state of the key.
Closes the **`change key state UI`**.	Removes the **`change key state UI`**.

END STATE

- The key's state is changed.

ALERT or ERROR STATES *(Requires a unique UI state or a message.)*

- The key has been corrupted.
- The key's state can't be changed for some reason.

ISSUES

- The states that the user can change needed to be identified.
- Brocade may have additional states that can be changed.
- When a user changes a key's state, does it have any effect? For example when a user changes a key's state to *compromised*, does that change of state get communicated to the client?

muc612 :: Export keys for emergency software decryption

PURPOSE: Enables a user to export data encryption keys to a password-protected file when the OpenKey client has failed and it's the only way to get the keys to the software decryption module.

INITIAL STATE

- An OpenKey client has failed.
- An LKM has the client's data encryption keys.
- A quorum of recovery cards and available.

FLOW

The User...	The System...
Starts by either:	
• Selecting the LKM that has OpenKey client's keys.	Shows the LKM as selected.
• Selecting the dataset (or datasets) encrypted with the keys.	Shows the dataset as selected.
Chooses the **`export for software decryption`**	Displays the **`export for software decryption UI`**.

command.	
	Asks the user to enter and confirm a password for the exported file.
Enters a password.	
Enters the password a second time to confirm it.	Compares the passwords to make sure they match.
	If the passwords don't match:
	• Displays an error message.
	• Enables the user to enter the passwords again.
	If the passwords match:
	• Displays a `save` dialog box.
	• Displays a default name for the file.
Can change the name of the file.	Displays the new name of the file.
Navigates to the folder where he wants to save the file.	Displays the contents of the folder the user has chosen in the dialog box.
Clicks the `export` button.	Displays the `authorization UI`.
The user is the recovery officer:	
Inserts her recovery card.	Reads the card.
Enters her password.	Verifies the password.
	Prompts for the next recovery card.
The process is repeated until the quorum is reached.	Exports the keys to the file.
	Saves the file in the folder the user has chosen.
	Removes the `export for software decryption UI`.

END STATE

- The data decryption keys that the software decryption module needs have been saved in the password-protect file with the name the user has given it in the folder the user has selected.

ALERT or ERROR STATES *(Requires a unique UI state or a message.)*

- Typical file-saving errors (e.g., a file with the same name already exists).

ISSUES

- This approach may also be an insecure way to share keys with a third party.

muc613 :: Audit a key's trail

PURPOSE: Enables the user to trace a key's whereabouts (when generated, archived, shared, recovered, destroyed)

INITIAL STATE

- The LKM is *ready*.

FLOW

The User...	The System...
Find dataset from the list of datasets	Queries the logs for activities on selected key ID with respect to the LKM selected and displays any successful and failed attempts of the following:
	• Generated when by the LKM
	• Backed up when and where to
	• Distributed when and to whom (from which LKM to which LKM, or to which client)
	• Export to file (from which LKM to which directory and filename)
	• Retrieved/used when and by which client
	• Retired when and by which admin
	• Rotated when and by which admin, replaced by which new key

	• Deleted when
Can save it as a report file	Hides UI
Closes UI	Hides UI

END STATE

- Same as initial state.

ALERT or ERROR STATES *(Requires a unique UI state or a message.)*

- None

ISSUES

- Export format to be determined (Rob/Gaurav)

muc614 :: Share keys with a trusted, but unlinked LKM (on the same network)

PURPOSE: Enables the user to share keys that are on one LKM with another LKM when the two have a trust relationship, but aren't linked.

INITIAL STATE

- The two LKMs are ready *ready*.
- The two LKMs have a trust relationship, but aren't linked (see muc012 - Create a trust relationship between two LKMs that aren't linked)
- The two LKMs are on a network that enables them to connect with each other.
- Dolphin automatically detects both LKMs.
- The user has found the keys she wants to share (see muc129 - Find an element in the system.

FLOW

The User...	The System...
Selects the keys she wants to share.	
Selects the LKM the keys are to be shared with.	
Chooses the `share keys` command.	Sends the keys to the selected LKM.
	Notifies the user that the keys have been shared successfully.
Acknowledges the message.	Removes the message.

END STATE

- The keys held by one LKM are now available to the other LKM.

ALERT or ERROR STATES *(Requires a unique UI state or a message.)*

- The keys aren't transmitted successfully.

ISSUES

- There's no procedure like this currently.

muc615 :: Share keys with a trusted, but unlinked LKM (on a different network)

PURPOSE: Enables a user to share keys that are on one LKM with another LKM when the two have a trust relationship, but aren't linked, and can't communicate over a network.

INITIAL STATE

- The two LKMs are ready *ready*.
- The two LKMs have a trust relationship, but aren't linked (see muc012 - Create a trust relationship between two LKMs that aren't linked)
- The two LKMs can't communicate over a network.

- The user has found the keys she wants to share (see muc129 - Find an element in the system).

FLOW

The User...	The System...
Selects the keys she wants to share.	
Chooses the **`share keys`** command.	Displays the **`share keys UI`**.
Chooses the trusted LKM she wants to share the keys with.	
Enters a password for the file and confirms it.	Records the password.
Clicks **`share`**.	Displays a Save dialog box.
Navigates to the folder where he wants to save the file and clicks **`save`**.	Saves the file in the folder the user has selected.
	Encrypts the file so only the trusted LKM can decrypt it.
	Protects the file with the password.

END STATE

- The keys are contained in an encrypted, password protected file.

ALERT or ERROR STATES *(Requires a unique UI state or a message.)*

- No keys have been selected.
- The file can't be saved.
- The password and the confirmation don't match.
- The password doesn't conform to the rules.

ISSUES

- Is the password necessary if the file is encrypted in a way that only the target LKM can decrypt it?
- The user who receives the file ought to be able to use the procedure in Import data encryption keys from a file to import it. Even better she out to be able to just drag-and-drop it onto the LKM icon.

Ready means:

- An LKM appliance is connected to the network.
- It's turned on.
- It's online.
- It's initialized.
- It's been added to Dolphin so the user can manage it.
- The user is logged into it.

-- JeffreySchwamberger - 19 May 2008

r18 - 26 Jun 2008 - 10:12:29 - JeffreySchwamberger