Inventarisierung DB Backend 2017

Ein Produkt der

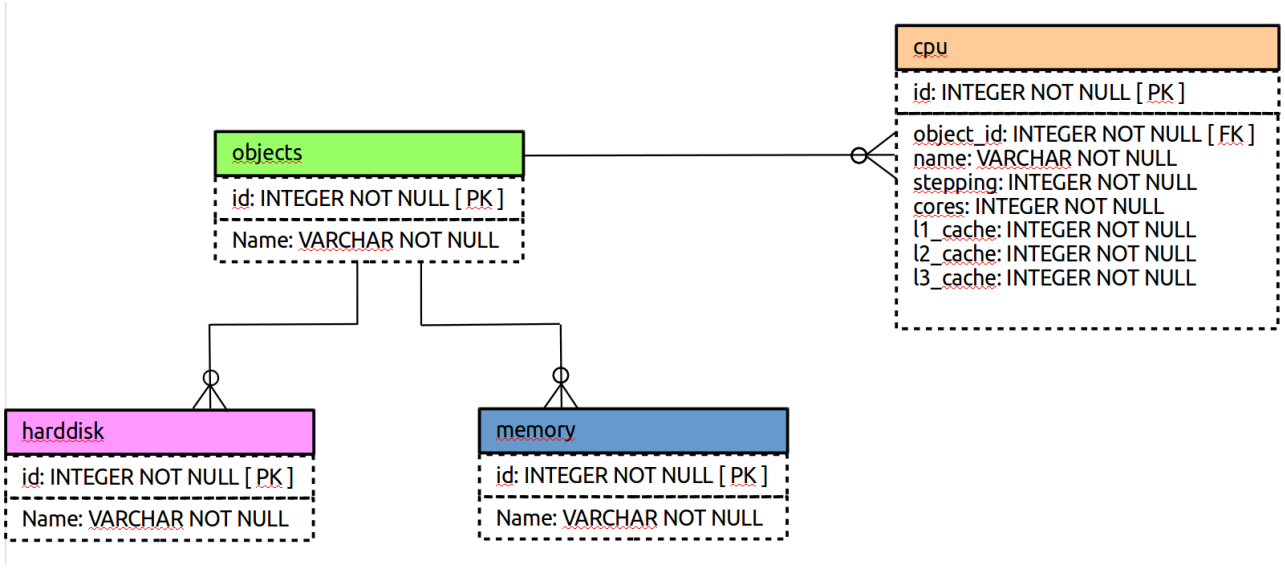
CAPE IT gmbh

Inhaltsverzeichnis

Installation

Voraussetzungen

- KIX Professional 17 - aktuelle Version
- Eine externe Datenbank mit Inventory-Daten organisiert in einer Tabellenmatrix, siehe unten folgende Abbildung



Dieses Bild ist nur ein Schema-Beispiel, um Ihnen zu zeigen, wie die Daten organisiert werden müssen, um mit diesem "DB"-Backend arbeiten zu können. Sie können so viele Tabellen mit so vielen Attributen haben, wie Sie möchten. Sie müssen aber immer einen foreign key (Fremdschlüssel) zu der Tabelle mit allen Objekten (bspw. Server, Mobile Devices,...) haben. Diese so genannte "object table" ist die primäre Objektliste für InventorySync und darf nur eine Reihe pro Objekt enthalten. Alle anderen Tabllen enthalten die spezifischen Objektdaten.

Installation

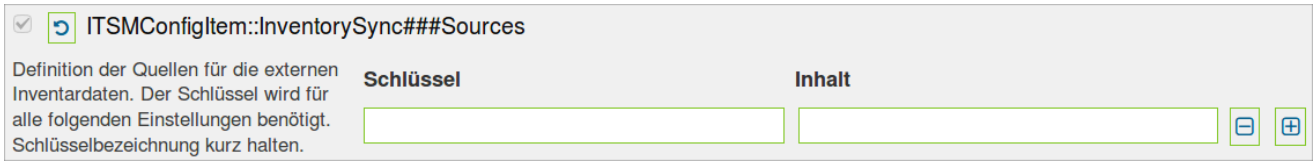
Installieren Sie das Paket *"kixpro-inventorysync-db"* mit der *"Paketverwaltung"* (Menü *"Admin"* Bereich *"Systemverwaltung"*) oder über die Kommandozeile.

Konfiguration

Das Paket beinhaltet eine Beispielkonfiguration, die Sie nach Ihren Bedürfnissen anpassen können. Um die Einstellungen für diese Funktion zu konfigurieren, wählen Sie in der SysConfig die Gruppe *"KIX Professional"*. Nachdem die Seite neu geladen wurde, wählen Sie die Untergruppe *"ITSMConfigItem::InventorySync"*.

Sources

- SysConfig-Einstellung: *ITSMConfigItem::InventorySync###Sources*



Eine Quelle benötigt einen benannten Kommunikationspunkt. Zur Konfiguration wird ein Hash benutzt. Der Hash-Schlüssel ist der interne Name der Quelle. bspw. *"opsisrv1"*. Der Hash-Wert ist der Anzeigename für diese Quelle, bspw. *"inventory1"*.

Sie können mehrere Quellen für jeden externen Inventory-Server hinzufügen.

Source-Backend-Mapping

- SysConfig-Einstellung: *ITSMConfigItem::InventorySync###Backend*

Diese Einstellung definiert das zu nutzende Backend. Zur Konfiguration wird ein Hash benutzt. Der Hash-Schlüssel ist der Identifikator der Quelle (in unserem Beispiel *"inventory1"*). Der Hash-Wert muss mit "DB" angegeben werden, damit das Modul InventorySync das opsi-Backend für diese Datenquelle benutzt.

Konfiguration in Config.pm

Nur der Name (und der interne Schlüssel) einer DB-Datenquelle und ihr Mapping auf die DBBackend wird in der SysConfig konfiguriert. Alle anderen "komplexen" Konfigurationen für die konfigurierten DB-Datenquellen erfolgt in der Config.pm auf eine Art und Weise, wie sie von der Konfiguration von CustomerUser-Backends bekannt ist.

Für jede DB-Datenquelle, die Sie in der SysConfig für InventorySync konfiguriert haben, müssen Sie einen Konfigurations-Hash in der Config.pm erstellen. Hier ist ein Beispiel für eine Konfiguration unserer Beispieldatenquelle.

Hier wird die Konfiguration für die ConfigItem-Klasse *"Computer"* vorgenommen, die ihre Daten aus einer weiteren MySQL-Datenbank *"ocsweb"* auf localhost erhält. Alle Objekte (Computer) sind eindeutig in der Datenbanktabelle (oder View) *"all_clients"* enthalten und der Bezeichner für jeden Client ist das Attribut *"HARDWARE_ID"* in dieser Tabelle.

Der Name des Objekts ist im Tabellenattribut *"NAME"* enthalten. Das als Objektkennung benannte Attribut (hier *"HARDWARE_ID"*) muss in allen anderen Tabellen/Views, aus denen das DB-Backend seine Daten beziehen soll, vorhanden sein (Foreign Key).

Gemeinsame Tabellen für Software (SW) und Hardware (HW) Informationen werden so konfiguriert, wie zwei separate/spezielle Tabellen/Views, die verwendet werden sollen, um alle Daten zu erhalten, die für die Gerätetypen *"PROZESSOR"* und *"CONTROLLER"* relevant sind.

Im Abschnitt *"Mapping"* werden die Attribute aus den konfigurierten Tabellen/Views in die Datenstruktur gemappt., die für jedes Objekt angelegt wird. Daher wird die Struktur für das Attribut *"InventoryContentXPath"* in der ConfigItem-Klassen definiert.

```
$Self -> {InventorySyncDB} -> {inventory1} = {
    Params => {
        # if you want to use an external database, add the
        # required settings
        # DSN => 'DBI:odbc:yourdsn',
        DSN => 'DBI:mysql:database=ocsweb;host=localhost',
        User => 'Test',
        Password => 'xxxxx',
        SourceCharset => 'utf-8',
        DestCharset => 'utf-8',
    },
    Classes => {
        Computer => {
            Params => {
                # client list and key value
                ObjectTable => 'all_clients',
                ObjectID => 'HARDWARE_ID',
                ObjectName => 'NAME',
                # common SW and HW tables/views
```

```
            # where our hardware/software info comes from
            HWTable => 'csweb.test_view2',
            SWTable => 'ocsweb.test_view1',
            # special tables to use for PROCESSOR/CONTROLLER mapping
            SpecialTables => {
                HW => {
                    PROCESSOR => 'cpu_table',
                    CONTROLLER => 'controllers',
                }
            }
        },
        # the following mapping defines the structure for
        # InventoryContentXpath
        Mapping => {
        HW => {
            PROCESSOR => {
                name => 'NAME',
                type => 'TYPE',
                version => 'VERSION',
            },
            serialNumber => {
                serialnumber => 'SNR',
            },
            BIOS => {
                description => 'DESCRIPTION',
                name => 'NAME',
            },
            CONTROLLER => {
                description => 'DESCRIPTION',
                name => 'NAME',
                type => 'CONTROLLER_TYPE',
            },
        },
        SW => {
            Software => {
                name => 'NAME',
                version => 'VERSION',
                descriptions => 'DESCRIPTION',
            },
        }
    }
}
```

}

Inventory Content XPath

Die Struktur und der Inhalt der Inventory-Daten (Hardware und Software), die vom DB-Backend zurückgegeben werden, wird in der "Config.pm" definiert. Der XPath hat allgemein folgende Struktur:

Hardware Xpath:

HW/<some device key>/<attribute>

Software Xpath:

SW/<configured key>/<attribute>