

cPanel Database Mapping

Abstraction of the Account-Database Relationship

Prepared for: cPanel, Inc. Partners & Customers

Prepared by: David Neimeyer, Quality Assurance and Integrations

April 14, 2010

Document Revision 12: October 28, 2010

Table of Contents

Executive Summary	1
What is Database Mapping?	1
Benefits of Database Mapping	1
Drawbacks of Database Mapping	1
When Is Database Mapping Available	1
Just the Facts	2
No Really, What is Database Mapping?	2
Okay, What is Database Prefixing?	2
Who's Affected by Database Mapping?	2
How Do I Enable and Disable DB Mapping?	3
Preamble	3
Where and How	3
Do I Need Database Mapping?	3
What About Version Downgrades?	3
Preamble	3
What to Expect	3
Weighing the Options	4
Restoring User Backups and cPanel Transfers	5
Transfers from Non-cPanel Systems	5
Longer Account Usernames	5
Supporting a Decoupled Primary Database User	5

Inherent Pitfall of Transfers	5
Technical Underpinnings	7
The cPanel User Model	7
cPanel 11.26 and Earlier	7
cPanel 11.28 and Later	7
YAML Datastore	8
MySQL	9
PostgreSQL	9
Cpanel::DB::Map	9
cpanel/bin/dbstoregrants	9
cpanel/bin/dbindex	10
cpanel/bin/mysqluserstore	10
Backups	10
Transfers	11
Pkgacct	11
Downgrade Safeguard for Transferred Accounts	11
Minor Back-merge into 11.25.0 for “dormant” YAML Datastore	11
Notification Helper for 11.28 Transfers	12
API Calls	12
cPAddons	12
Implementation Details	13
Database Resource Naming Convention	13
Before Database Mapping	13
11.28 with Prefixing On	13
11.28 with Prefixing Off	13
Setting cPanel Prefix State	14
CLI: dbmptool Usage	14

Summary and Purpose	14
Specifying Multiple --type Parameters	15
Specifying --dbusers and --dbs Parameters	15
cpanel/bin/setupdbmap	15
Flag file /var/cpanel/version/11.25.1	15
Account Modification	16
Application Changes	16
WHM	16
cPanel	17
Command Line Interface	18
Third-Party Applications	19
Security	19
Troubleshooting and Tips	20
First Tip: Don't downgrade unless it's sane	20
Second Tip: Test DB Prefixing "Off" for your environment	20
Third Tip: Secure your implementation	20
Glossary	22
Terms	22
Appendix A	24
User Model	24
A1: cpuser Model without DB Mapping	24
A2: cpuser Model with DB Mapping	25
Appendix B	26
YAML Datastore	26
B1: Samples of /var/cpanel/databases/\$cpuser.yaml	26
B2: Sample /var/cpanel/databases/grants_\$cpuser.yaml	28
B3: Sample /var/cpanel/databases/dbindex.db	29

B4: Sample /var/cpanel/databases/users.db	30
Appendix C	31
Grants	31
C1: MySQL grants without DB Mapping	31
C2: MySQL grants with DB Mapping and DB Prefixing turned on	31
C3: MySQL grants with DB Mapping and DB Prefixing turned off	32
Appendix D	33
Cron Tasks	33
D1: Cron tasks added with the introduction of DB Mapping	33
Appendix E	34
API Calls	34
E1: API calls whose parameters are affected by DB Prefixing	34
Appendix F	35
dbmaptool	35
F1: Sample Usage for Adding a Database	35
F2: Sample Usage for Adding a Database Virtual User	35
F3: Sample Usage for Adding a List of Databases and a List of Database Virtual Users	36
Document Change Log	37
Revision 12	37
Product Version Number	37
DB Mapping Feature Set	37

Executive Summary

What is Database Mapping?

Database Mapping (DB Mapping) is the process by which cPanel & WHM maps the relationship between cPanel accounts and their database user, database virtual users, and databases. The DB Mapping code is responsible for Database Prefixing and assists with Account Transfer mechanisms.

Benefits of Database Mapping

The benefits of DB Mapping revolve around two key points:

- Account-database relationships are preserved when transferred from non-cPanel accounts.
- Database usernames can be the optimal length.

Drawbacks of Database Mapping

The drawbacks of DB Mapping fall into two categories:

- cPanel versions prior to 11.28 do not have DB Mapping. This makes downgrading to 11.26 laborious, at best. User accounts that use DB Mapping will need to remain on an 11.28 system.
- Third-party integrations or customizations that provide their own tools for database information and administration may not work with accounts using DB Mapping. If the integration utilizes the cPanel-provided APIs, however, there should be few or no complications.

When Is Database Mapping Available

DB Mapping is available in all builds of cPanel & WHM 11.28 and later. It will not be back-merged into previous versions.

Just the Facts

No Really, What is Database Mapping?

Any web hosting environment must have a mechanism responsible for verifying which accounts have access to which databases, which database users they administer, and, of course, the credentials they use to authenticate. The most conventional way to do this is by establishing a namespacing paradigm that revolves around a user-based prefix.

A quick example account might have an account name *'myacct'* with a prefix *'myacct_'*. At account creation, the web hosting environment enforces the namespace convention, and a relational trust is established between account name *'myacct'* and the prefix *'myacct_'*

DB Mapping is simply a different method of defining that relationship: an alternate solution to namespacing.

Okay, What is Database Prefixing?

As stated in [No, Really, What is Database Mapping](#), many web hosting environments, including cPanel, use a database prefix (DB Prefixing) to designate which database resources belong to which account. cPanel uses the account name followed by an underscore. If an example account named *'myacct'* were to create a database with the name *'somedb'*, the resulting database would actually be *'myacct_somedb.'*

Database users created and administered by the account would also have the prefix. DB Mapping indexes the ownership of database resources as they are created, and thus removes the requirement of prefixing resource names.

Who's Affected by Database Mapping?

Everyone who uses cPanel version 11.28 and beyond is affected by DB Mapping because it is a core change in cPanel. All previous cPanel functionality should behave the same until the cPanel administrator performs the Prefixing 'Opt-Out'. After 'Opt-Out,' the cPanel users will gain the benefit of creating databases that do not have *'myusername_'* as a prefix.

Additionally, any new database virtual users will not have the *'myusername_'* prefix either, allowing for names that utilize the maximum length of characters—16 in a standard MySQL install.

How Do I Enable and Disable DB Mapping?

Preamble

DB Mapping refers to the cPanel code base. DB Mapping is factored into cPanel 11.28. Technically, it cannot be turned on or off. Mapping of information happens regardless of any UI form submission or `/var/cpanel/cpanel.config` value. DB Prefixing is the behavior that is affected by the setting/variable change.

By disabling DB Prefixing in the *SQL Services* area of WHM, the user alters the UI and code paths used by the cPanel interface and API calls. This allows cPanel systems an ‘Opt-Out’ design with respect to the behaviors available in 11.28.

Where and How

Within WHM’s *SQL Services*, you will find *Disable Database Prefix*. This form provides a *Process* button that will “Opt-Out” of database prefixing. To remove the prefix restriction, check the confirmation box and click *Process*. No existing data will be modified by the change**.

Disabling Prefixing is a one-way action. Once this form is submitted, the options will not be available again. Additionally, this action will commit the cPanel instance to maintaining a code base that has DB Mapping—namely, 11.28 and later.

**In cPanel versions prior to 11.26, all cPanel accounts have a wildcard grant in MySQL related to their database prefixes. This wildcard grant is removed when Prefixing is ‘Off’ and present when Prefixing is ‘On’.

Do I Need Database Mapping?

Not everyone needs the features offered in cPanel’s DB Mapping. These features are available whenever their functionality is required. Their use is voluntary. The only caveat is that cPanel accounts that use these features cannot be transferred or downgraded to a cPanel system versioned below 11.28.

What About Version Downgrades?

Preamble

In version 11.28, cPanel has changed at its core. Downgrading an 11.28 install (with user accounts that depend on or utilize a sans-prefix environment) to 11.26 would not be wise and, at this time, is not a valid course of action, except in the most dire of circumstances. 11.26 has no concept of DB Mapping or accounts whose database usernames differ from their account usernames, or a sans-prefix ownership of databases or database virtual users.

What to Expect

DB Mapping features that perform Mapping dependencies also inform the cPanel instance that downgrading is not feasible. If a cPanel install with Mapping dependencies attempts a downgrade, the downgrade process will not complete—it will immediately terminate. The downgrade cannot be forced by any programmatic means within cPanel.

Explicit MySQL grants are made in 11.28 (from normal MySQL end-user action) that will be retained in the downgrade process. However, the relationship of those grants to the cPanel user account dissipates in the 11.26 environment. For example, in 11.28, if a cPanel account is created with the username *'someone'* and DB Prefixing has been disabled so that database resources are not required to be prefixed, any databases made by *'someone'* will have grants for the database. At downgrade, however, the relationship between cPanel account *'someone'* and any databases that are not named like *'someone_'* is invalid. The grant will persist and ownership of the resource is still available via a MySQL client or phpMyAdmin. The resources are merely neither exposed by the cPanel UI nor accessible by the cPanel 11.26 code.

The 11.26 UI and functions will only be looking for databases with grants for the db username *'someone'* AND a name like *'someone_%'* when the user is authenticated as *'someone.'* Likewise, any database virtual users will not be associated with the cPanel account unless they are appropriately prefixed.

Weighing the Options

Is It Sane?

- If the cPanel system in question was only upgraded to 11.28 and downgraded back to 11.26, then no changes in user data have occurred. Downgrade is sane.
- If the cPanel system in question, while utilizing the 11.28 code base, did not 'Opt-Out' of Prefixing, and no transfers from non-cPanel accounts were performed, then all user data generated during the 11.28 lifespan was modeled using the 11.26 paradigm, i.e., Prefixing is in place and utilized. Downgrade is sane.

Tough Row to Hoe

Downgrading to a cPanel version below 11.28 after capitalizing on the DB Mapping feature set will dissolve the account-database relationships established therein. The data will persist and be directly accessible, but only outside the cPanel interface and code base.

An example would be a custom install of site software. The site software would have the literal, valid credentials necessary to operate. However, those databases would not be accessible via the cPanel interface for administration.

If one had no other choice but to downgrade to version 11.26, any account that utilizes the DB Mapping feature set would need to be coerced into the standardized prefixed model:

- Username and database username need to be equal.
- Any database virtual user will need to be prefixed.
- Any databases will need to be prefixed.
- All third-party applications will need to have the new username (or virtual username) and new database name properly placed within their authentication/configuration mechanisms.
- The wildcard grant for the cPanel user prefix should be present in MySQL.

That Sad Moment

Sometimes hard decisions need to be made. Properly weighing one's options before 'Opt-Out' can save lots of frustration. Nothing beyond voluntary deletion of data is beyond salvation and cPanel Support is available 24 hours, 7 days a week.

Restoring User Backups and cPanel Transfers

As mentioned earlier, DB Mapping is hard coded into cPanel 11.28. User account backups generated with cPanel 11.28 may contain database resources that depend on DB Mapping and should only be restored on cPanel 11.28 and later. cPanel-to-cPanel transfers must be treated in the same regard.

This is not to say that it's not *plausible* to move an 11.28 account to an 11.26 host. Instead, it's only *feasible* to transfer a cPanel 11.28 account to a version of cPanel which has DB Mapping.

Since DB Mapping is aware of the traditional cPanel user model, transferring or restoring an 11.26 account to a cPanel host with DB Mapping is valid and handled gracefully.

Transfers from Non-cPanel Systems

DB Mapping provides the necessary abstraction for translating non-cPanel user models into cPanel's user model. The obvious benefit is that non-cPanel database relations stay the same.

Longer Account Usernames

cPanel enforces an 8-character account username maximum. DB Mapping exposes an exception to this soft limit account username restriction, up to 16 characters.

Supporting a Decoupled Primary Database User

cPanel 11.28 does not support a fully decoupled relationship between the cPanel account and the primary database user (dbowner). When transferring an account from a non-cPanel system which has a unique dbowner, the import process will create a virtual database user, named after the unique dbowner, and place all respective grants to that new virtual database user. This behavior provides persistent relationship integrity between the resources and for the management of those resources.

Inherent Pitfall of Transfers

The exception of longer usernames is critical when performing a "mirrored" transfer. However, if a transferred account has a long username, and DB Prefixing is enabled on the hosting cPanel instance, the long username will be used as the account prefix for all new database resources. Therefore, an inherent pitfall is introduced into the traditional cPanel user model.

Any cPanel virtual database username created while Prefixing is "On" must contain the account username followed by an underscore character. If the transferred account has an exceptionally long username, it's possible to prematurely reach the MySQL hard limit of 16-character database usernames. The traditional cPanel user model, with the soft limit of 8 characters for account usernames, guarantees that the prefix for database resources never exceeds 9 characters. Therefore, virtual database usernames will always have up to 7 characters to delineate themselves from other database users. If the cPanel account username that is transferred is 15 characters long, there will be no characters left when the cPanel user attempts to create a virtual database user if Prefixing is "On."

Technical Underpinnings

The cPanel User Model

The cPanel User Model outlined here is concerned with database resource creation and modification. This includes SQL databases and database (virtual) users. The creation of tables, row insertions, and other data manipulation data for a given database are inherent to the SQL RDBMS, and beyond the scope of this document.

cPanel 11.26 and Earlier

cPanel versions earlier than 11.28 have a simple user model. Each cPanel account is uniquely identified by the account username. Since the username is unique on any given cPanel installation, it is used as the basis for all named database resources. Each named resource is given a prefix, *\$username_*.

For example, when the account *someone* of *cpanelenduser.net* creates a database *db1*, the result is a database literally named *someone_db1*. The same behavior is exhibited for user-created database users. If *someone* creates a database user named *john*, the result is *'someone_john.'*

All code requests for SQL actions are granted based on the fulfillment of one of two criteria:

- An explicit SQL grant exists, with the proper privileges, for the requesting database user and the database.
- The requesting database user is performing an action that creates, deletes, or modifies a database resource whose name begins with the requesting username followed by an underscore.

[Appendix A1](#) illustrates this relationship.

cPanel 11.28 and Later

cPanel 11.28 introduces DB Mapping to the user model. Unlike 11.26 and earlier, database resource permission is not dependent on the literal cPanel account username. The cPanel account is still uniquely identified by the username; however, any database resource created or available to the account is itemized outside of the SQL RDBMS.

Each cPanel account has a YAML file located in `/var/cpanel/databases/` that is responsible for itemizing the account-database relationship for that cPanel user (cpuser). See [YAML Datastore](#) for more information.

If example account *someone* requests the creation of a database named *db1*, cPanel will attempt to create a database with the literal name *db1*. Likewise, any creation request for a database virtual user by *someone* will be generated as it is literally passed within the request.

All code requests for SQL actions are granted based on the fulfillment of the following criteria:

- During resource creation, a SQL resource with the same name cannot already exist in the RDBMS.
- During resource creation, the SQL resource name cannot match select cPanel-reserved names.
- Existing resources must be itemized within ownership scope, as defined in the cpuser's YAML datastore.

All requests for SQL actions made with RDBMS tools other than those provided by cPanel products will not directly modify the account-database relationship; the datastore will not reflect any fulfilled action.

[Appendix A2](#) illustrates this relationship.

YAML Datastore

YAML is a human-friendly data serialization standard for all programming languages (reference: www.yaml.org). Each cPanel account has a YAML file located in `/var/cpanel/databases/` which is named respective to the cPanel account username; i.e., `$username.yaml`. A binary cache of each YAML file also exists within the same directory; i.e., `$username.cache`.

The YAML file contains a scope for the MySQL and PostgreSQL RDBMSs. Within each SQL scope are properties which define the cPanel account-database relationship. Defined properties include:

- owner
 - * The primary SQL user for the cPanel account.
- server
 - * The system IP of the server hosting the SQL resource.
- dbs
 - * Databases for which the primary SQL user has full privileges.
- dbusers
 - * SQL users maintained by the primary SQL user.
 - * Each user listed can have a nested scope of properties, similar to the primary SQL user's scope.

An example YAML file is provided in [Appendix B1](#).

MySQL

cPanel control of MySQL functionality related to DB Mapping is unchanged in 11.28. MySQL transactions originating from the cPanel UI and code base are performed as the MySQL *root* user. These actions are facilitated by a secure, escalated-privilege code path. All third-party applications and site-installed software, however, perform actions as either the primary database user or a virtual database user.

The cPanel functions responsible for interfacing with MySQL are dependent on packages in the *Cpanel::DB::Map* namespace. See [Cpanel::DB::Map](#) for more details.

In cPanel 11.28, databases created through cPanel insert a MySQL GRANT in *mysql.db*, itemizing that the primary database user has ALL privileges for the new database. Previous versions of cPanel do not make this table insert, but instead rely on a wildcard grant that is generated during the cPanel account creation process. [Appendices C1, C2 and C3](#) illustrate the difference between these MySQL grant structures.

PostgreSQL

PostgreSQL functionality, like MySQL, remains unchanged in 11.28. PostgreSQL actions originating from the cPanel UI and code base are performed as the *postgres* user, in the same secure, escalated privilege mode as the MySQL *root* user. All third-party applications and site-installed software, however, perform actions as either the primary database user or a virtual database user.

The cPanel functions responsible for interfacing with PostgreSQL are dependent on packages in the *Cpanel::DB::Map* namespace. See [Cpanel::DB::Map](#) for more details.

All grants are created on a database object (table, column, view, sequence, database, foreign-data wrapper, foreign server, function, procedural language, schema, or tablespace) and the concept of wildcard grants does not exist in PostgreSQL.

Cpanel::DB::Map

The Perl namespace *Cpanel::DB::Map* instantiates a Perl object used by cPanel SQL modules. It functions as a fluent interface for pre-existing data storage and caching mechanisms present in the *Cpanel* namespace. These packages are the linchpin for reading and writing the account-database relationship datastores.

cpanel/bin/dbstoregrants

dbstoregrants is a utility script (binutil) for recording SQL grants for individual cPanel accounts. It will create a file called *grants_\${cpuser}.yaml* in */var/cpanel/databases/*, where *\$cpuser* is the cPanel account name. Because the data stored

within these files contains private SQL user data, it's imperative that they remain only readable to the root user. If SQL grants need to be restored in an RDBMS, `cpanel/bin/restoregrants` will read `grants_$cpuser.yaml` and perform that functionality. See [Appendix B2](#) of an example of `grants_$cpuser.yaml`.

`dbstoregrants` is executed at database resource creation, deletion, and when a host access is granted or revoked for a database resource.

cpanel/bin/dbindex

`dbindex` is a binutil which generates `/var/cpanel/databases/dbindex.db`. `dbindex.db` is an index of databases and the respective cPanel accounts to which they belong. See [Appendix B3](#) for an example of `dbindex.db`.

`dbindex` utilizes `/etc/dbowners`, an index of primary database users and the respective cPanel accounts to which they belong.

`dbindex` is executed by a cron job prior to importing a transfer account. See [Appendix D1](#).

cpanel/bin/mysqluserstore

`mysqluserstore` is a binutil for recording host and user MySQL information for indexing purposes. Data is stored as a YAML structure in `/var/cpanel/databases/users.db`. See [Appendix B4](#) for an example of `users.db`.

`mysqluserstore` is executed at database user creation and deletion. Additionally, `mysqluserstore` has a cron job. See [Appendix D1](#).

Backups

System backups are not affected by DB Mapping. All system backup utilities function identically to previous cPanel & WHM behavior. The `Cpanel` modules used by `/scripts/cpbackup` do not directly traverse any code path introduced by DB Mapping.

User-generated backups via the cPanel interface or API1 `Fileman::fullbackup` determine the cpuser's database resources using the same modules, scripts, and administrative binaries that perform ordinary UI and API utilities. Therefore, the `Cpanel::DB::Map` namespace will be used at various points in code execution.

User backup files do not contain any fundamental changes. Additional information is present in the account configuration that itemizes the primary database username. Also, the backup will contain the account's DB Mapping YAML datastore.

A user backup generated on a system with DB Mapping, restored onto a cPanel host which does not contain DB Mapping, is likely to complete without error. However, database resources may not be available to the cPanel account until the hosting system is updated to a version of cPanel containing the DB Mapping code.

Transfers

cPanel 11.28 introduces three functional changes to WHM transfers.

Pkgacct

During transfer from a non-cPanel server, the remote account must be packaged for transfer. Acquisition of the remote account's resources must be performed specific to the hosting environment from which the account is being migrated. In 11.28, a sync mirror is utilized to retrieve the correct environment package script, which is bundled with other necessary scripts. The sync mirror guarantees the remote host the most up-to-date version of the *pkgacct* script.

Previous versions of cPanel relied on the cPanel host to provide the environment package script.

WHM still offers an *Override* function, documented at <http://docs.cpanel.net/twiki/bin/view/AllDocumentation/WHMDocs/CopyAccount>, for cPanel administrators or cPanel Technicians to supersede the standard bundle with custom scripts.

Downgrade Safeguard for Transferred Accounts

During the transfer process for non-cPanel accounts, a flag file is generated. The purpose is to denote that there are cPanel accounts on the host system that rely on DB Mapping. While this flag file exists, */scripts/upcp* will preempt any attempt to downgrade to a version of cPanel which does not include DB Mapping. See [Flag file /var/cpanel/version/11.25.1](#) for more information.

Minor Back-merge into 11.25.0 for “dormant” YAML Datastore

As of build 46156 of cPanel 11.25.0, *Transfers.pm* and */scripts/pkgacct* contain code logic related to DB Mapping which helps negotiate the import of cPanel accounts created on cPanel 11.28. The code is only responsible for storing the cpuser YAML datastore in */var/cpanel/databases/*, which will not be utilized or modified by the host system. The presence of the datastore can be useful later, in two scenarios:

- When the host system is upgraded to a cPanel version with DB Mapping, the datastore will be updated.
- If the cPanel account is migrated back to a cPanel host with DB Mapping, the datastore will be present in the transfer process.

Caution must be asserted, as a full back-merge of DB Mapping is not present. A caveat for the above scenarios exists when the cPanel administrator modifies the account name. */scripts/modcpuser* and the respective WHM interface will not alter the “dormant” datastore filename or contents, and therefore the datastore is disassociated from the cpuser.

Notification Helper for 11.28 Transfers

The cPanel 11.28 transfer interface in WHM offers a warning system for host-account incompatibility. If the administrator attempts to import an account containing database resources that are not prefixed and the host system is configured for DB Prefixing, a warning will be displayed. The administrator then has the opportunity to cancel the transfer or process due to the obvious incompatibility.

API Calls

SQL API calls now depend on the *Cpanel::DB::Map* namespace. Calls that return information about database resources reference cPanel user YAML files located in */var/cpanel/databases/*. Calls that create, delete or modify database resources will update the cPanel user YAML and perform the necessary SQL action. Furthermore, *cpanel/bin/dbstoregrants* will execute for all resource creation, deletion or modification; *cpanel/bin/mysqluserstore* will execute for database virtual user creation and deletion. See the respective sections in [Technical Underpinnings](#) for more details.

All API calls that create a database resource must take a name for the new resource. If the cPanel system has DB Prefixing turned off, the resource name will be taken literally and will not be prefixed at creation time. See [Appendix E1](#) for a full list of SQL API calls which exhibit this behavior.

cPAddons

CPAddons core functionality has not changed with the introduction of DB Mapping. However, it is important to note the database resource creation behavior that occurs when initializing a cPAddon for a cPanel account. During the setup process, if any database resource is created, its name will be prefixed, regardless of the state of DB Prefixing. CPAddon database resources will always be prefixed based on the downer name. See [Appendix B1](#) for an example cpuser YAML.

Implementation Details

Database Resource Naming Convention

Before Database Mapping

Versions of cPanel prior to 11.28 do not contain DB Mapping. For any version of cPanel without DB Mapping, all ownership of database resources is governed by the name of the resource. When a database resource is created, a prefix is added, formatted as *\$cpuser_*, where *\$cpuser* is the cPanel account username.

11.28 with Prefixing On

cPanel systems with DB Mapping which have DB Prefixing enabled, (the default value) exhibit previous cPanel behavior for naming conventions. The prefix is determined by the dbowner, the primary database username associated with the cpuser. By convention, the primary database username is identical to cpuser, the cPanel account username.

Example

cpuser *someone*, on host *cpanelenduser.net*, wishes to create a MySQL database named *aDatabaseName*. The cpuser logs into cPanel and navigates to <https://cpanelenduser.net:2083/frontend/x3/sql/index.html>. Here, the cpuser can *Create a New Database* using the input box. In the input field, he would type *aDatabaseName* and click the *Create Database* button. The resulting MySQL database is *someone_aDatabaseName*. See [Appendix C1](#) for MySQL grant details.

11.28 with Prefixing Off

cPanel systems with DB Mapping which have DB Prefixing disabled will create database resources based on the literal name requested. Without prefixing, any database resource name with valid characters will be permitted, barring those already used by a preexistent database resource or reserved system name. See [Appendix C2](#) for MySQL grant details.

Example

cpuser *someone*, on host *cpanelenduser.net*, wishes to create a MySQL database named *aDatabaseName*. The cpuser logs into cPanel and navigates to <https://cpanelenduser.net:2083/frontend/x3/sql/index.html>. Here, the cpuser can *Create a New Database* using the input box. In the input field, she types *aDatabaseName*, then she clicks the *Create Database* button. The resulting MySQL database is *aDatabaseName*. See [Appendix C3](#) for MySQL grant details.

Setting cPanel Prefix State

DB Prefixing is determined by the `database_prefix` value defined in `/var/cpanel/cpanel.config`.

- A value of “1” denotes that DB Prefixing is enabled and will emulate prefixing behavior prior to DB Mapping.
- A value of “0” denotes that DB Prefixing is disabled and all database resources created by cPanel accounts will be created literally as requested.
- The absence of a `database_prefix` definition will behave as if a value of “1” existed.

Manually modifying `cpanel.config` is HIGHLY DISCOURAGED and can produce unexpected behaviors. WHM provides the necessary UI for setting DB Prefixing to “0,” “Off”. See the *WHM* section of [Application Changes](#) for details.

CLI: dbmapprool Usage

Summary and Purpose

`/usr/local/cpanel/bin/dbmapprool` is a binutil command for modifying the account-database relationship as itemized in a cpuser’s YAML file. Modifications are limited to the assignment of database resources. Databases can be assigned to the dbowner, or database virtual users can be assigned to the dbowner.

Assignment of databases to virtual users and removal of itemized resources are not functions of this binutil. All values provided for database resource names are literal and will not be affected by the state of DB Prefixing.

`dbmapprool` is only responsible for modifying the the cpuser’s YAML file and will not perform any SQL action.

Parameter	Required	Usage
<code>\$cpuser</code>	Yes	Name of the cPanel account to receive changes. First argument to <code>dbmapprool</code> ; no switch flag.
<code>--type \$type</code>	Yes	<code>\$type</code> must be either <code>mysql</code> or <code>pg</code> . If specified multiple times, the last one overrides previous definitions.
<code>--dbusers \$dbusers</code>	No	<code>\$dbusers</code> is a comma-separated list of virtual users to assign to the dbowner; the list should be enclosed in quotation marks.
<code>--dbs \$dbs</code>	No	<code>\$dbs</code> is a comma-separated list of databases to assign to the dbowner; the list should be enclosed in quotation marks.

Table 4.1: Catalog of available parameters, their requirement status, and usage. See [Appendix F](#) for examples.

Specifying Multiple --type Parameters

Only one *type* should be provided in a single execution of *dbmapprool*. If multiple *type* parameters are provided, the last one will be used.

Specifying --dbusers and --dbs Parameters

Multiple parameters of either *dbusers* or *dbs* will behave identically to multiple parameters of “type;” only the last switch will be honored.

Both *dbusers* and *dbs* parameters may be provided in a single execution. However, *dbmapprool* will not associate the two parameters. Instead, it treats the parameters as two distinct executions, once for each parameter switch.

See [Appendix F](#) for examples of usage and the effect on a *cpuser* YAML file.

cpanel/bin/setupdbmap

/usr/local/cpanel/bin/setupdbmap is a binutil whose primary responsibility is creating *cpuser* YAML files when DB Mapping is applied to a cPanel system for the first time. It utilizes the *Cpanel::DB::Map* namespace to create */var/cpanel/database/\$cpuser.yaml*, where *\$cpuser* is cPanel account username. See [YAML Datastore](#) for details concerning the information parsed and stored.

Its secondary purpose is to ensure that *Cpanel::DB::Map* and the other various *Cpanel* modules with DB Mapping dependencies have access to cPanel account-database information. Since *setupdbmap* is responsible for the integrity of the account-database information, it's possible that it may create or alter datastore structures and schemas appropriately for use by *Cpanel* modules and various binutils. Therefore, *setupdbmap* is performed at each execution of */scripts/upcp*. There should be no need to execute *setupdbmap* manually, since new cPanel functionality, or need for new datastores, only arises when updating via execution of */scripts/upcp*.

Flag file /var/cpanel/version/11.25.1

The existence */var/cpanel/version/11.28* will preempt any attempt to downgrade cPanel via */scripts/upcp* to a cPanel version without DB Mapping.

Two scenarios will *touch* this flag file:

- Transferring a non-cPanel account to cPanel with DB Mapping, and the account contains database resources that are not prefixed.
- *Disable Database Prefix* in WHM's *SQL Services*.

Account Modification

Modifying a cPanel account username in cPanel prior to DB Mapping performs the following in regard to SQL services:

- Changes the SQL username to equal the cPanel account username.
- Modifies the prefix of all existing database resources to reflect the new SQL username.

If the cPanel system has DB Mapping, the following also occurs:

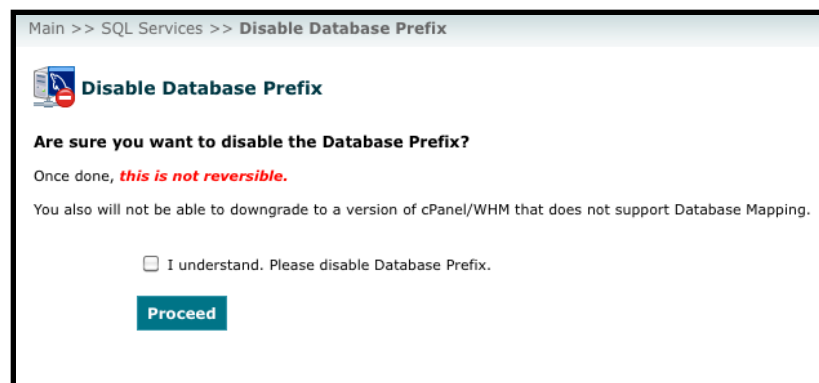
- cpuser's YAML file is renamed.
- The information within the YAML file is updated to reflect changes made to the dbowner and any affected database resource.
- `grants_$cpuser.yaml` is renamed.
- `dbindex.db` is updated.
- `users.db` is updated.
- `/etc/dbowners` is updated.

Application Changes

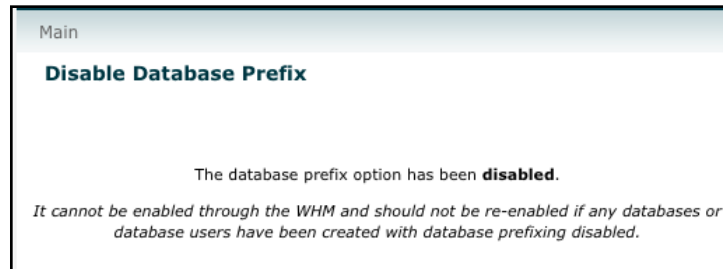
WHM

SQL Services >> Disable Database Prefix

cPanel 11.28 implements the DB Prefixing “OFF” state as a one-time occurrence. This is controlled by the existence of a flag file, `/var/cpanel/version/11.25.1`. Turning off Prefixing will touch this flag file. A value of “0” will be defined for “`database_prefix`” in `/var/cpanel/cpanel.config`.



Screenshot 2.1: WHM's SQL Services provides a one-time Disable Database Prefix option



Screenshot 2.2: This notification appears after Disable Database Prefix is performed.

SQL Services >> Change MySQL DB Owner Password

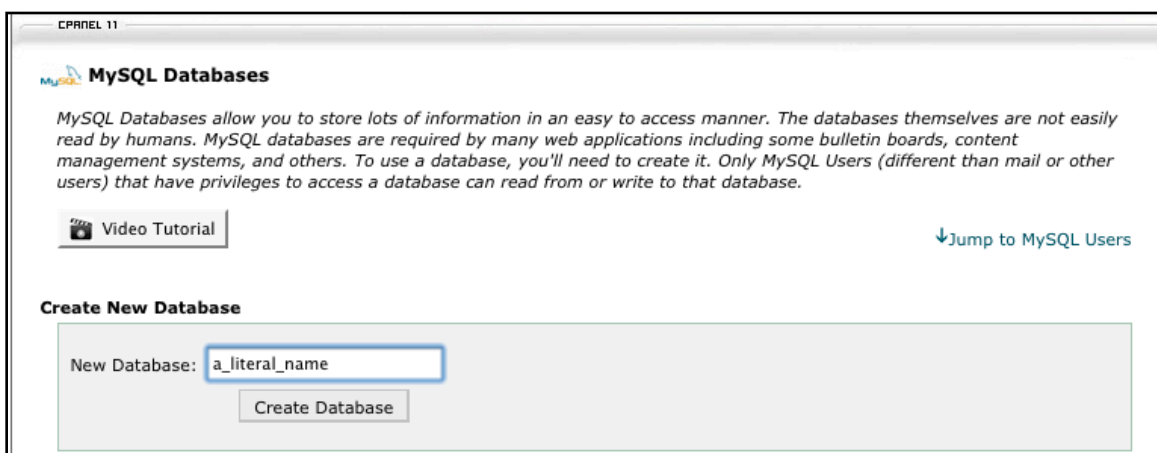
This function was previously named *Change MySQL User Password*. The scope of this function has been narrowed to only allow the administrator to change the password for a cPanel account's dbowner. System MySQL users, or any other MySQL user not claimed as a dbowner by a cPanel account, will not appear in the drop-down selection element.

cPanel

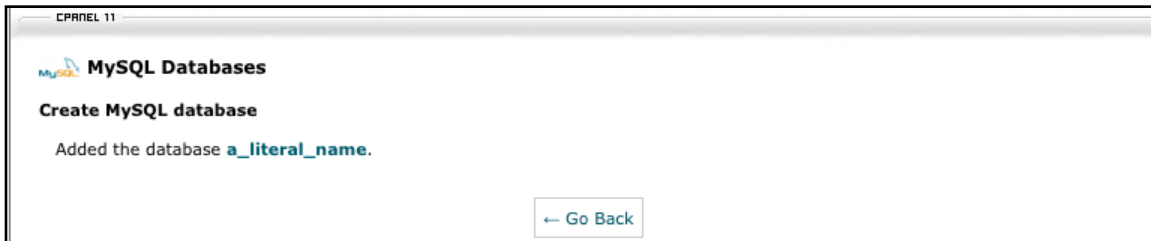
Databases >> MySQL and PostgreSQL

The various cPanel SQL management UIs offered within the *Database* section of the main cPanel UI have been updated to accommodate DB Prefixing in the "On" or "Off" state. If prefixing is on, all UIs render identically to cPanel versions without DB Mapping; that is, input fields are prepended, visually, with prefixes, and database resource names are given prefixes at the point of creation.

If prefixing is off, all UI input fields for database resource creation are presented bare; that is, no visual prefix appears before the text input field, and JavaScript validation allows for maximum character lengths. All data posted in the request is taken literally at the point of resource creation.



Screenshot 3.1: Users may provide an arbitrary name for database creation when DB Prefixing is off.



Screenshot 3.2: A database created with an arbitrary name with DB Prefixing off.

Command Line Interface

Added

The following scripts have been added for DB Mapping:

- `/scripts/killmysqlwildcard`
 - * Helper script invoked when disabling Prefixing. Removes system-created MySQL wildcard grants.
- `/scripts/check_users_my_cnf`
 - * Helper script for validating cpusers' `.my.cnf` files, which are located in each cpuser's home directory.
- `/usr/local/cpanel/bin/setupdbmap`
 - * Invoked during execution of `/scripts/upcp`. Initializes cpuser YAML datastores for the cPanel account.
- `/usr/local/cpanel/bin/dbmaptool`
 - * Binutil for adding database resources to a cpuser YAML datastore.
- `/usr/local/cpanel/bin/mysqluserstore`
 - * Stores host and user MySQL information in `/var/cpanel/databases/users.db`.
- `/usr/local/cpanel/bin/dbindex`
 - * Indexes databases and their cPanel account in `/var/cpanel/databases/dbindex.db`.
- `/usr/local/cpanel/bin/dbstoregrants`
 - * Stores SQL user grants in `/var/cpanel/databases/grants_$(cpuser).yaml`, where `$(cpuser)` is the cPanel account username.

Modified

The following scripts have been modified for use with DB Mapping and incorporate the use of `Cpanel::DB::Map`:

- `/scripts/upcp`
- `/scripts/wwwacct`
- `/scripts/modcpuser`
- `/scripts/mysqlup`
- `/scripts/dropmysqldb`

- `/scripts/suspendmysqlusers`
- `/scripts/unsuspendmysqlusers`
- `/scripts/update_db_cache`

Removed

The following scripts have been removed in cPanel 11.28:

- `/scripts/mysqldeluserdb`
- `/scripts/mysqladduserdb`
- `/scripts/mysqldelusers`
- `/scripts/removeuserdb`

Third-Party Applications

cPanel's patches for phpMyAdmin and phpPgAdmin have been modified to use the dbowner name for authentication. Prior versions of these patches used the cpuser name.

Security

`/var/cpanel/databases/` by design is world-readable, but not world-writable, and owned by the root user and group (e.g., `drwxr-xr-x 2 root root 4096`). All cpuser YAML datastores, `dbindex.db`, and `users.db` also adhere to this permission and ownership design. cpuser grants in the YAML datastore, however, are only readable and writable by the root user.

As discussed in [Troubleshooting and Tips](#), the current implementation of DB Mapping does not enforce a security policy on the contents of cpuser YAML datastores. As with previous versions of cPanel, the cPanel utilities responsible for performing SQL requests enforce a context-appropriate security policy to ensure the integrity of system resources and cpuser account-database relationships. Therefore, standard cPanel SQL usage does not compromise pre-existing functionality.

Troubleshooting and Tips

First Tip: Don't downgrade unless it's sane

Downgrading to a cPanel system that utilizes DB Mapping to a version that does not can compromise the account-database relationship for cPanel accounts. These inherent limitations are clearly itemized in [Just The Facts](#). All technical knowledge required to bypass cPanel safeguards is present in this document. Leveraging this knowledge is done “at your own risk.”

Second Tip: Test DB Prefixing “Off” for your environment

Operating cPanel systems with Prefixing “Off” can introduce an artificially frequent occurrence of “name collision” when creating cPanel accounts and cpuser database resources. Because arbitrary names are allowed, the probability that common names are requested will increase and therefore the denial to use those names will also increase.

Keep in mind, DB Prefixing is a option exposed by DB Mapping. The primary goal of DB Mapping is a fluent interface for RDBMSs in cPanel, non-cPanel transfers and, eventually, cPanel clustering.

Third Tip: Secure your implementation

Security is always a primary concern and responsible administration will avoid exposing user and system data unnecessarily. Migrating accounts presents certain inherent risks, and you should always be conscious about what information you're placing on the host system.

The cpuser YAML datastore, like any system data file, can be crafted to exploit a system's trust in its own data. A single YAML datastore can arbitrarily expose database resources for any given cpuser on the host system for the attacker. One possible vector for such an attack is available through account restoration wherein the packaged account to be restored contains a tainted cpuser YAML datastore. The restore process in cPanel 11.28 has been altered in two ways to reduce the potential risk and the effectiveness of such malevolence.

- Grants are sanitized before inserting into MySQL.
- Databases are restored as the cPanel user, and not the root user.

Likewise, manual manipulation of the cpuser YAML datastore, however well intentioned, can lead to unexpected and potentially dangerous situations if improperly or erroneously performed.

Glossary

Terms

Binutil

A system utility script located in `/usr/local/cpanel/bin/`. For system use only.

cPanel Account Username

Given name for the account used by cPanel on behalf of the **cpuser**.

cpuser, User

A cPanel account user.

Database Resource

A database or database user.

Database Primary User, dbowner

The database user used by cPanel on behalf of the **cpuser**.

Database Virtual User, db virtuser

A database user created by the **cpuser**.

Datastore

File containing information which adheres to a defined schema. In the context of DB Mapping, a **YAML** file.

DB Mapping

A feature set available in cPanel version 11.28 and beyond. **DB Mapping** allows for the use arbitrary **database resource** names associated with a cPanel account. **DB Mapping** negates the requirement of **DB Prefixing**.

DB Prefixing

A naming convention for **database resources** based on the cPanel account username. For example, a database named `db1` for **cpuser someone** exists in the RDBMS as `someone_db1`.

dbuser

Within the context of WHM or system scripts located in `/scripts/`, **dbuser** is synonymous with **dbowner**. Within the context of a **cpuser** **YAML** datastore, **dbuser** is synonymous **database virtual user**.

Flag file

A empty file generated to reflect a state or change of state in the host system environment.

Opt-in

Elect to participate in a referenced method or paradigm. In the context of **DB Prefixing**, one can opt into exposure of **DB Mapping**'s use of arbitrary **database resource** names.

Opt-out

Elect to not use a referenced method or paradigm. In the context of **DB Prefixing**, one can opt out of exposure of **DB Mapping**'s use of arbitrary **database resource** names.

Owner

Within the context of a **cpuser** YAML datastore, **owner** is synonymous with **dbowner**.

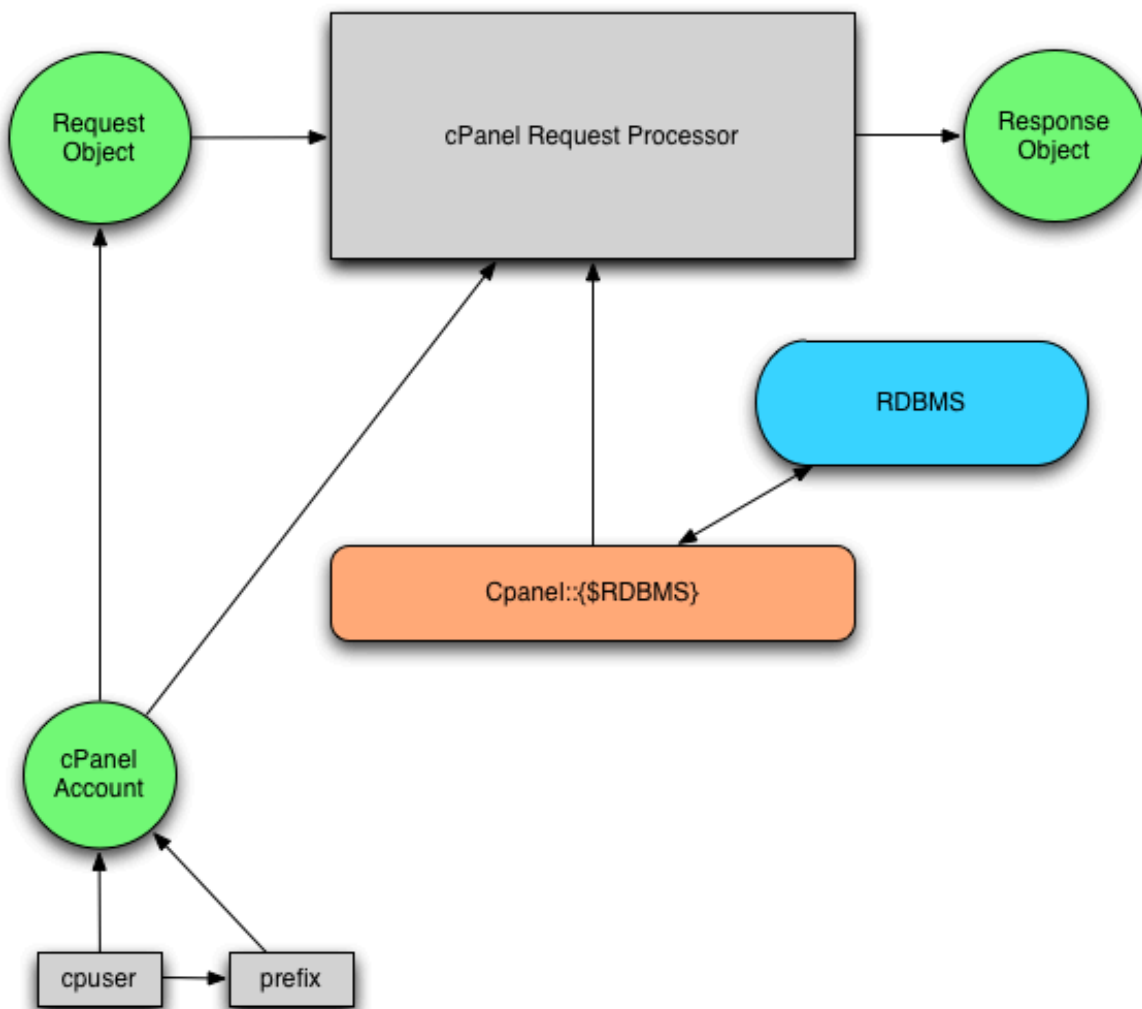
YAML

Human-friendly data serialization standard for all programming languages (reference: www.yaml.org).

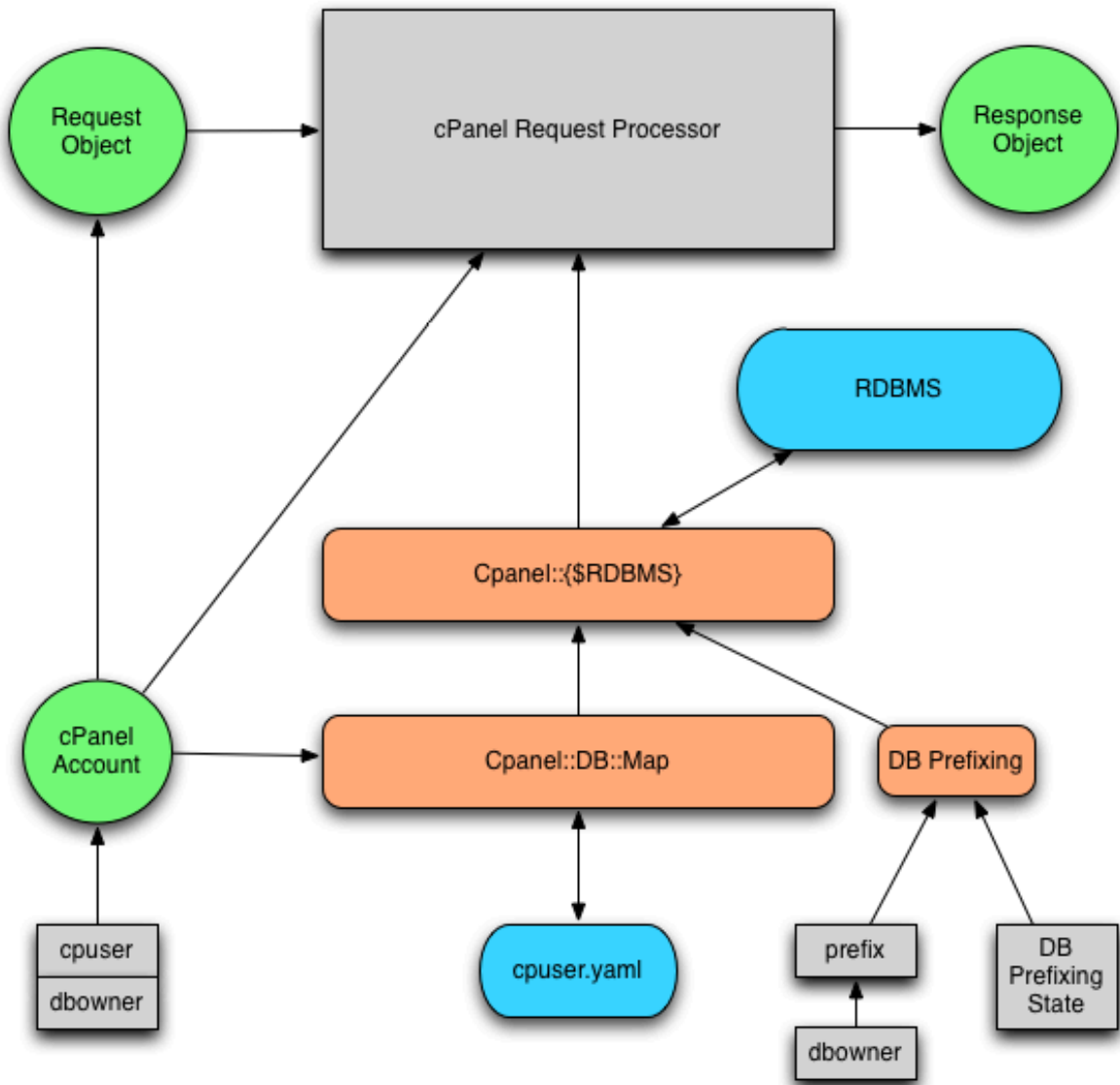
Appendix A

User Model

A1: cpuser Model without DB Mapping



A2: cpuser Model with DB Mapping



Appendix B

YAML Datastore

B1: Samples of `/var/cpanel/databases/$cpuser.yaml`

Simple

cpuser *someone* on system IP *10.1.1.1*. Note the *owner* field, the *dbowner*, is the same as the cpuser. cpuser has two databases: *someone_db1* and *someone_db2*. cpuser has one database virtual user, *someone_lilone*, which has privileges for the *someone_db1* database.

```
---
MYSQL:
  dbs:
    someone_db1: 10.1.1.1
    someone_db2: 10.1.1.1
  dbusers:
    someone_lilone:
      dbs:
        someone_db1: 10.1.1.1
      server: 10.1.1.1
  owner: someone
  server: 10.1.1.1
```

“Simple” example above, with Prefixing turned off and database resources added

Same account itemized above, but a database *noprefixexample* and a database virtual user *fullengthuser* were added after the cPanel administrator disabled Prefixing.

```
---
MYSQL:
  dbs:
    someone_db1: 10.1.1.1
    someone_db2: 10.1.1.1
    noprefixexample: 10.1.1.1
  dbusers:
    someone_lilone:
      dbs:
        someone_db1: 10.1.1.1
      server: 10.1.1.1
    fulllengthuser:
      dbs: {}
      server: 10.1.1.1
  owner: someone
  server: 10.1.1.1
```

cPAddon

New account example from above. *someone* installed a WordPress cPAddon, specifying that a new database and database virtual user named *wp1* be created during initialization. Note that cPAddon resources are always prefixed.

```
---
MYSQL:
  dbs:
    someone_wp1: 10.1.1.1
    someone_db1: 10.1.1.1
    someone_db2: 10.1.1.1
    noprefixexample: 10.1.1.1
  dbusers:
    someone_lilone:
      dbs:
        someone_db1: 10.1.1.1
        server: 10.1.1.1
    fulllengthuser:
      dbs: {}
      server: 10.1.1.1
    someone_wp1:
      dbs:
        someone_wp1: 10.1.1.1
        server: 10.1.1.1
  owner: someone
  server: 10.1.1.1
```

B2: Sample `/var/cpanel/databases/grants_$cpuser.yaml`

Sample `grants_someone.yaml` from “Simple” example in Appendix B1.

```

---
MYSQL:
  someone:
    someone:
      - Grants for someone@localhost
      - GRANT USAGE ON *.* TO 'someone'@'localhost' IDENTIFIED BY PASSWORD
        '*mysqlpasswdhash'

      - GRANT ALL PRIVILEGES ON `someone`.* TO 'someone'@'localhost'
      - GRANT ALL PRIVILEGES ON `someone\\_db1`.* TO 'someone'@'localhost'
      - GRANT ALL PRIVILEGES ON `someone\\_db2`.* TO 'someone'@'localhost'
      - GRANT ALL PRIVILEGES ON `someone\\_%`.* TO 'someone'@'localhost'
    someone_lilone:
      - Grants for someone_lilone@localhost
      - GRANT USAGE ON *.* TO 'someone_lilone'@'localhost' IDENTIFIED BY PASSWORD
        '*mysqlpasswdhash'

      - GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, REFERENCES, INDEX, ALTER,
        CREATE TEMPORARY TABLES, LOCK TABLES, CREATE ROUTINE ON `someone\\_db1`.* TO
        'someone_lilone'@'localhost'
    someone_lilone:
      - Grants for someone_lilone@localhost
      - GRANT USAGE ON *.* TO 'someone_lilone'@'localhost' IDENTIFIED BY PASSWORD
        '*mysqlpasswdhash'

      - GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, REFERENCES, INDEX, ALTER,
        CREATE TEMPORARY TABLES, LOCK TABLES, CREATE ROUTINE ON `someone\\_db1`.* TO
        'someone_lilone'@'localhost'

```

B3: Sample `/var/cpanel/databases/dbindex.db`

The cPanel system has three users: *acct1*, *acct2*, and *someone*. cpuser *acct1* has no databases. cpuser *acct2* has one database, *acct2_db1*. cpuser *someone*, which was created after the cPanel administrator turned Prefixing off, has two databases: *db1* and *db2*. cpuser *someone* has assigned privileges for database virtual user *//someone* to database *db1*.

NOTE:

- cpuser *acct1* does not have any databases. The account will not be in this database index.
- cpuser *someone* only has databases indexed against the cpuser name, as that is the purpose of this YAML datastore.

```

---
MYSQL:
  acct2_db1: acct2
  db1: someone
  db2: someone

```

B4: Sample `/var/cpanel/databases/users.db`

cPanel system `10.1.1.1` resolves to `050.cpanel.test` and has the same account and database resources as the example in Appendix B3. Additionally, the cPanel administrator has added a custom MySQL test user, `testuser`, which has MySQL access from any host.

NOTE: This YAML datastore itemizes MySQL users, which includes system, primary database users, virtual users, and custom.

```

---
"%":
  testuser: 1
050.cpanel.test:
  root: 1
127.0.0.1:
  root: 1
localhost:
  acct1: 1
  acct2: 1
  someone: 1
  lilsomeone: 1

```

Appendix C

Grants

The following SHOW GRANTS statements illustrate the individual MySQL database grants created by cPanel for a cpuser. In examples C1 and C2, the cpuser *someone* has requested the creation of two databases, *db1* and *db2*.

C1: MySQL grants without DB Mapping

Because a wildcard grant exists, it is unnecessary for cPanel to itemize each database for which the cpuser has access.

```
mysql> show grants for 'someone'@'localhost';
+-----+
| Grants for someone@localhost
+-----+
| GRANT USAGE ON *.* TO 'someone'@'localhost' IDENTIFIED BY PASSWORD '*passwordhash' |
| GRANT ALL PRIVILEGES ON `someone`.* TO 'someone'@'localhost' |
| GRANT ALL PRIVILEGES ON `someone_%`.* TO 'someone'@'localhost' |
+-----+
```

C2: MySQL grants with DB Mapping and DB Prefixing turned on

DB Mapping will always add a grant to the dbowner for each database created.

```
mysql> show grants for 'someone'@'localhost';
+-----+
| Grants for someone@localhost
+-----+
| GRANT USAGE ON *.* TO 'someone'@'localhost' IDENTIFIED BY PASSWORD '*passwordhash' |
| GRANT ALL PRIVILEGES ON `someone`.* TO 'someone'@'localhost' |
| GRANT ALL PRIVILEGES ON `someone_%`.* TO 'someone'@'localhost' |
| GRANT ALL PRIVILEGES ON `someone\_db1`.* TO 'someone'@'localhost' |
| GRANT ALL PRIVILEGES ON `someone\_db2`.* TO 'someone'@'localhost' |
+-----+
```

C3: MySQL grants with DB Mapping and DB Prefixing turned off

Taking the example from C2, if the cPanel administrator turns Prefixing off, then the prefix wildcard grant is removed for all MySQL accounts. The cpuser is now free to create databases with arbitrary names. In this example, *someone* created a database named *a_literal_name*.

```
mysql> show grants for 'someone'@'localhost';
+-----+
| Grants for someone@localhost
+-----+
| GRANT USAGE ON *.* TO 'someone'@'localhost' IDENTIFIED BY PASSWORD '*passwordhash' |
| GRANT ALL PRIVILEGES ON `someone`.* TO 'someone'@'localhost' |
| GRANT ALL PRIVILEGES ON `a_literal_name`.* TO 'someone'@'localhost' |
| GRANT ALL PRIVILEGES ON `someone_db1`.* TO 'someone'@'localhost' |
| GRANT ALL PRIVILEGES ON `someone_db2`.* TO 'someone'@'localhost' |
+-----+
```

Appendix D

Cron Tasks

D1: Cron tasks added with the introduction of DB Mapping

```
30 */2 * * * /usr/local/cpanel/bin/mysqluserstore >/dev/null 2>&1
15 */2 * * * /usr/local/cpanel/bin/dbindex >/dev/null 2>&1
```

Appendix E

API Calls

E1: API calls whose parameters are affected by DB Prefixing

API	Module	Function	Affected Parameters	New Scope
API1	Mysql	adddb	dbname	If Prefixing is off, <i>dbname</i> will be taken literally. No prefix will be added at creation.
API1	Mysql	adduser	username	If Prefixing is off, <i>username</i> will be taken literally. No prefix will be added at creation.

All other API calls that use a database resource as a parameter require the literal name, regardless of DB Mapping or the state of DB Prefixing.

Appendix F

dbmaptool

dbmaptool allows the cPanel administrator to add arbitrary database resource names to a cpuser YAML datastore. All examples below are made for cpuser *someone*. The cpuser's dbowner is *someone*. The account is hosted on a server whose system IP is *10.1.1.1*.

For each sample, the cpuser's YAML file is cleared of any previous data to illustrate the exact effect of each executed command. Sample F3 provides a PostgreSQL example and requires PostgreSQL to be installed on the cPanel system.

F1: Sample Usage for Adding a Database

For cpuser *someone*, add a MySQL database *mydb*.

```
/usr/local/cpanel/bin/dbmaptool someone --type mysql -dbs 'mydb'
```

The resulting */var/cpanel/databases/someone.yaml* has *mydb* added to the *dbs* list within the *MYSQL* scope.

```
---
MYSQL:
  dbs:
    mydb: 10.1.1.1
  dbusers: {}
  owner: someone
  server: 10.1.1.1
```

F2: Sample Usage for Adding a Database Virtual User

For cpuser *someone*, add a MySQL database virtual user *lilsomeone*.

```
/usr/local/cpanel/bin/dbmaptool someone --type mysql -dbusers 'lilsomeone'
```

The resulting */var/cpanel/databases/someone.yaml* has *lilsomeone* added to the *dbusers* list within the *MYSQL* scope.

```

---
MYSQL:
  dbs: {}
  dbusers:
    l1someone:
      dbs: {}
      server: 10.1.1.1
  owner: someone
  server: 10.1.1.1

```

F3: Sample Usage for Adding a List of Databases and a List of Database Virtual Users

For cpanel user *someone*, add PostgreSQL databases *pgdb1* and *pgdb2*, plus add database virtual users *lilpg1* and *lilpg2*.

```
/usr/local/cpanel/bin/dbmapprool someone --type pg -dbs 'pgdb1,pgdb2' --dbusers 'lilpg1,lilpg2'
```

The resulting `/var/cpanel/databases/someone.yaml` has *pgdb1* and *pgdb2* added to the *dbs* list, plus *lilpg1* and *lilpg2* added to the *dbusers* list, all within the *PGSQL* scope.

```

---
MYSQL:
  dbs: {}
  dbusers: {}
  owner: someone
  server: 10.1.1.1
PGSQL:
  dbs:
    pgdb1: 10.1.1.1
    pgdb2: 10.1.1.1
  dbusers:
    lilpg1:
      dbs: {}
      server: 10.1.5.186
    lilpg2:
      dbs: {}
      server: 10.1.1.1
  owner: someone
  server: 10.1.1.1

```

Document Change Log

Revision 12

Product Version Number

This document has been update to reflect cPanel's new product version numbering schema and the public release number in which DB Mapping is available. cPanel test build version 11.25.1 also contains a pre-production copy of the DB Mapping code base.

Details about this schema are available on the cPanel website, <http://docs.cpanel.net/twiki/bin/view/AllDocumentation/InstallationGuide/CpanelProductVersions>

DB Mapping Feature Set

Previously published revisions of this document reference cPanel's 11.25.1 feature set of DB Mapping. Numerous changes have been made to that pre-production code. Those changes are itemized in brief below. Revision 12 of this document is the authoritative source of the publicly available DB Mapping code base as published in cPanel & WHM 11.28.

Errata and Redaction Summary

- dbowner is not a customizable field for use by an administrator or cPanel account
- WHM's account modification behavior has been simplified by enforcing the dbowner field as non-customizable
- WHM's *SQL Services* function for modifying the primary database user's password has been rebranded to reflect the underlying paradigm of the cPanel account-dbowner.