Acronis

Acronis Storage 2.4

Storage-as-a-Service Integration Guide

January 15, 2019

Copyright Statement

Acronis International GmbH, 2002-2016. All rights reserved.

"Acronis" and "Acronis Secure Zone" are registered trademarks of Acronis International GmbH.

"Acronis Compute with Confidence", "Acronis Startup Recovery Manager", "Acronis Active Restore",

"Acronis Instant Restore" and the Acronis logo are trademarks of Acronis International GmbH.

Linux is a registered trademark of Linus Torvalds.

VMware and VMware Ready are trademarks and/or registered trademarks of VMware, Inc. in the United States and/or other jurisdictions.

Windows and MS-DOS are registered trademarks of Microsoft Corporation.

All other trademarks and copyrights referred to are the property of their respective owners.

Distribution of substantively modified versions of this document is prohibited without the explicit permission of the copyright holder.

Distribution of this work or derivative work in any standard (paper) book form for commercial purposes is prohibited unless prior permission is obtained from the copyright holder.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Third party code may be provided with the Software and/or Service. The license terms for such third parties are detailed in the license.txt file located in the root installation directory. You can always find the latest up-to-date list of the third party code and the associated license terms used with the Software and/or Service at `http://kb.acronis.com/content/7696`

Acronis patented technologies

Technologies, used in this product, are covered and protected by one or more U.S. Patent Numbers: 7,047,380; 7,275,139; 7,281,104; 7,318,135; 7,353,355; 7,366,859; 7,475,282; 7,603,533; 7,636,824; 7,650,473; 7,721,138; 7,779,221; 7,831,789; 7,886,120; 7,895,403; 7,934,064; 7,937,612; 7,949,635; 7,953,948; 7,979,690; 8,005,797; 8,051,044; 8,069,320; 8,073,815; 8,074,035; 8,145,607; 8,180,984; 8,225,133; 8,261,035; 8,296,264; 8,312,259; 8,347,137; 8,484,427; 8,645,748; 8,732,121 and patent pending applications.

Contents

1. Introduction **1**

1.1 About This Guide 1

1.2 About WHMCS 1

1.3 About S3 Clusters 2

1.4 Integration Methods 4

2. Integration via Command-Line Interface (CLI) **5**

2.1 Managing S3 Users 5

2.1.1 Adding S3 Users 6

2.1.2 Listing S3 Users 6

2.1.3 Querying S3 User Information 7

2.1.4 Disabling S3 Users 7

2.1.5 Deleting S3 Users 7

2.1.6 Generating S3 User Access Key Pairs 8

2.1.7 Revoking S3 User Access Key Pairs 8

2.2 Managing S3 User and Bucket Limits via CLI 8

2.2.1 Setting Operations per Second for Users 8

2.2.2 Setting Bandwidth per Second for Users 9

2.2.3 Querying User Limits 9

2.2.4 Deleting User Limits 9

2.2.5 Setting Operations per Second for Buckets 10

2.2.6 Setting Bandwidth per Second for Buckets 10

2.2.7 Querying Bucket Limits 10

2.2.8 Deleting Bucket Limits 11

3. Integration via REST API **12**

3.1 Requirements for Integration via REST API . . . 12
3.1.1 Configuration . . . 13
3.1.2 Statistics . . . 14
3.2 Managing S3 Users and Listing Buckets via REST API . . . 14
3.2.1 Creating S3 Users . . . 14
3.2.2 Listing S3 Users . . . 15
3.2.3 Querying S3 Users . . . 15
3.2.4 Disabling S3 Users . . . 16
3.2.5 Enabling S3 Users . . . 16
3.2.6 Deleting S3 Users . . . 16
3.2.7 Generating S3 Access Keys . . . 17
3.2.8 Revoking S3 Access Keys . . . 17
3.2.9 Listing User Buckets . . . 17
3.3 Managing S3 User and Bucket Limits via REST API . . . 18
3.3.1 Setting Operations per Second for Users . . . 19
3.3.2 Setting Bandwidth per Second for Users . . . 19
3.3.3 Querying User Limits . . . 19
3.3.4 Deleting User Limits . . . 19
3.3.5 Setting Operations per Second for Buckets . . . 20
3.3.6 Setting Bandwidth per Second for Buckets . . . 20
3.3.7 Querying Bucket Limits . . . 20
3.3.8 Deleting Bucket Limits . . . 20
3.4 Obtaining Usage Statistics via REST API . . . 21
3.4.1 Listing Statistics Objects . . . 21
3.4.2 Querying Statistics Objects . . . 21
3.4.3 Deleting Statistics Objects . . . 22

4. Integration with WHMCS . . . 23
4.1 Requirements . . . 24
4.1.1 Configuration . . . 24
4.1.2 Includes . . . 25
4.1.3 Hooks . . . 28
4.1.4 Statistics . . . 36
4.2 Managing S3 Users in WHMCS . . . 37
4.2.1 Creating S3 Users . . . 38
4.2.2 Listing S3 Users . . . 39

4.2.3 Querying S3 Users . . . 41
4.2.4 Disabling S3 Users . . . 43
4.2.5 Enabling S3 Users . . . 44
4.2.6 Deleting S3 Users . . . 45
4.2.7 Generating S3 Access Keys . . . 46
4.2.8 Revoking S3 Access Keys . . . 48
4.3 Managing S3 User and Bucket Limits in WHMCS . . . 49
4.3.1 Setting User Limits . . . 50
4.3.2 Querying User Limits . . . 51
4.3.3 Deleting User Limits . . . 52
4.3.4 Setting Buckets Limits . . . 53
4.3.5 Querying Bucket Limits . . . 55
4.3.6 Deleting Bucket Limits . . . 56
4.4 Obtaining Usage Statistics in WHMCS . . . 57
4.4.1 Listing Statistics Objects . . . 57
4.4.2 Querying Statistics Objects . . . 59
4.4.3 Deleting Statistics Objects . . . 60

CHAPTER 1

Introduction

Acronis Storage is a software-defined storage optimized for storing large amounts of data. It provides data redundancy (replication and erasure coding), high availability, self-healing, and storage sharing.

1.1 About This Guide

This document will help you integrate Amazon S3 compatible services into your WHMCS provisioning and billing system. The guide is primarily intended for developers who already have working Acronis Storage clusters with properly configured Amazon S3-like roles and gateways.

In this document, you will find examples of integrating Acronis Storage S3 clusters via CLI and REST API as well as in WHMCS. Using this guide as a starting point, you will be able to create basic storage-as-a-service offerings based on Acronis Storage.

1.2 About WHMCS

WHMCS is an all-in-one hosting automation platform with client management, provisioning of services, billing and support. It handles everything from signup to termination of customers. Its functionality is expandable with extensions, add-ons, and hooks executing third-party code on certain events. You can find more information about WHMCS at https://www.whmcs.com/.

1.3 About S3 Clusters

Acronis Storage allows you to export cluster disk space to customers in the form of an S3-like object-based storage.

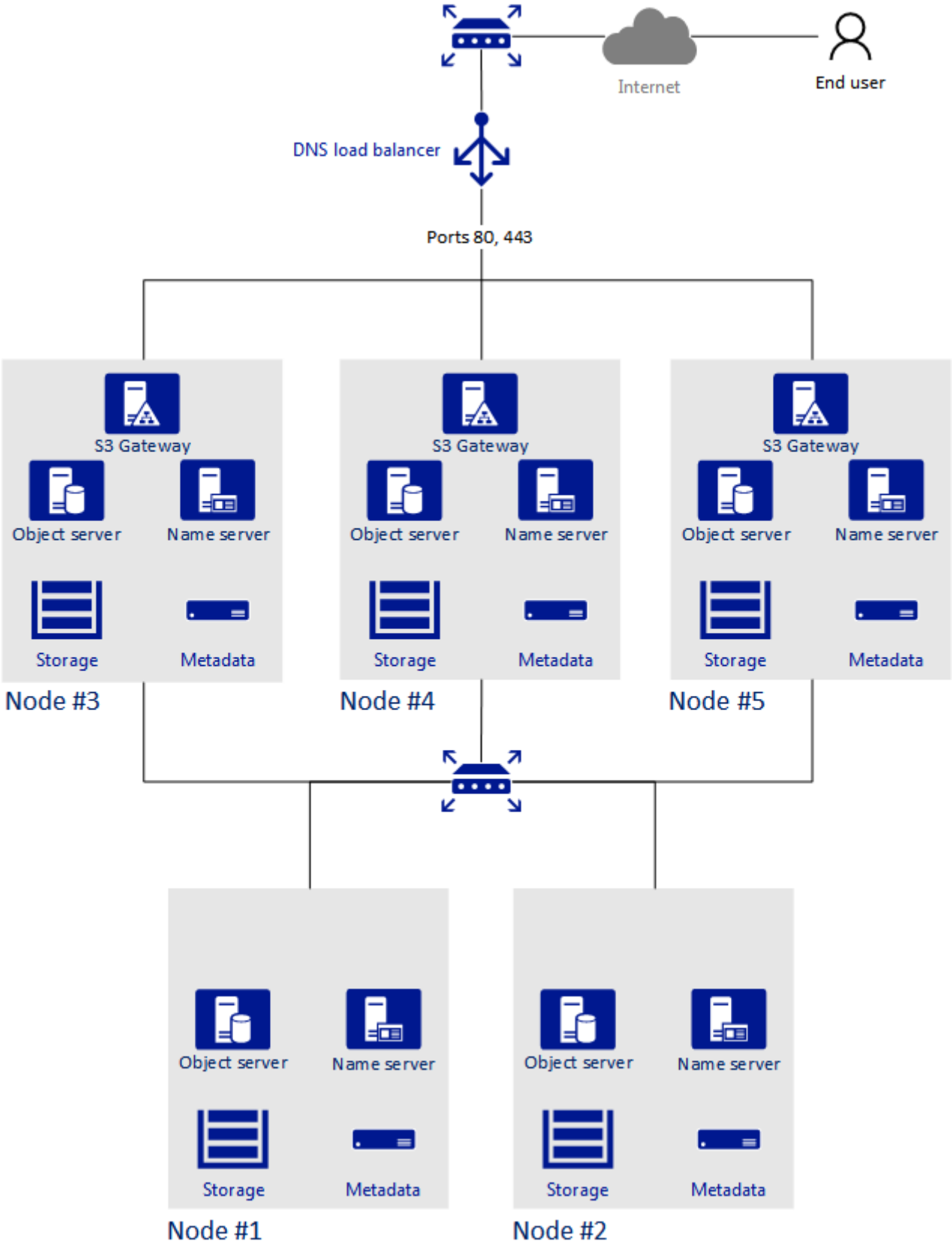
Acronis Storage is implemented as an Amazon S3-like API, which is one of the most common object storage APIs. End users can work with Acronis Storage as they work with Amazon S3. You can use the usual applications for S3 and continue working with it after the data migration from Amazon S3 to Acronis Storage.

The infrastructure of the object storage consists of the following entities: object servers (OS), name servers (NS), and the S3 gateways (GW).

These entities run as services on the Acronis Storage nodes. Each service should be deployed on multiple Acronis Storage nodes for high availability.

This section shows a sample object storage deployed on top of an storage cluster of five nodes that run various services. The final setup is shown on the figure below.

1.3. About S3 Clusters

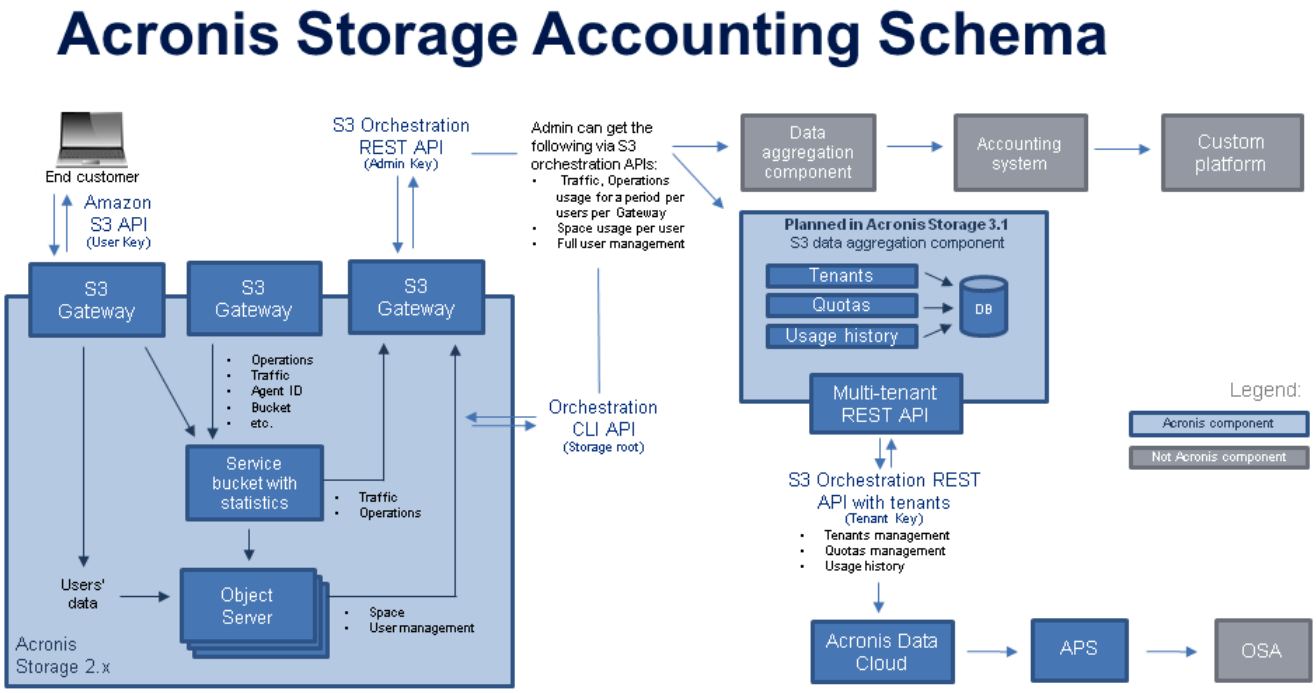


1.4 Integration Methods

Acronis Storage provides an orchestration representational state transfer (REST) API as well as an Amazon S3 compatible REST API.

With the orchestration API, you can manage users and buckets, configure user and bucket limits, and collect usage statistics. You can use the orchestration API by means of a single CLI tool shipped with Acronis Storage.

The Amazon S3 compatible REST API also enables you to manage users and buckets, configure user and bucket limits, and collect usage statistics. The user model and access policies comply with those of Amazon S3.



CHAPTER 2

Integration via Command-Line Interface (CLI)

This chapter explains ways to use the command-line interface to provision, enable, disable, and terminate S3 users as well as set user and bucket limits for billing purposes.

2.1 Managing S3 Users

The concept of S3 user is one of the base concepts of object storage along with those of object and bucket (container for storing objects). The Amazon S3 protocol uses a permission model based on access control lists (ACLs) where each bucket and each object is assigned an ACL that lists all users with access to the given resource and the type of this access (read, write, read ACL, write ACL). The list of users includes the entity owner assigned to every object and bucket at creation. The entity owner has extra rights compared to other users. For example, the bucket owner is the only one who can delete that bucket.

User model and access policies implemented in Acronis Storage comply with the Amazon S3 user model and access policies.

User management scenarios in Acronis Storage are largely based on the Amazon Web Services user management and include the following operations: create, query, and delete users as well as generate and revoke user access key pairs.

You can manage users with the `ostor-s3-admin` tool. To do this, you will need to know the ID of the volume that the users are in. You can obtain it with the `ostor-ctl get-config` command. For example:

```
# ostor-ctl get-config -n 10.94.97.195
VOL_ID             TYPE     STATE
0100000000000002   OBJ      READY
...
```

Note: As `ostor-s3-admin` commands are assumed to be issued by object storage administrators, they do not include any authentication or authorization checks.

2.1.1 Adding S3 Users

You can generate a unique random S3 user ID and an access key pair (S3 Access Key ID, S3 Secret Access Key) using the `ostor-s3-admin create-user` command. You need to specify a user email. For example:

```
# ostor-s3-admin create-user -e user@email.com -V 0100000000000002
UserEmail:user@email.com
UserId:a49e12a226bd760f
KeyPair[0]:S3AccessKeyId:a49e12a226bd760fGHQ7
KeyPair[0]:S3SecretAccessKey:HSDu2DA00JNGjnRcAhLKfhrvlymzOVdLPsCK2dcq
Flags:none
```

S3 user ID is a 16-digit hexadecimal string. The generated access key pair is used to sign requests to the S3 object storage according to the Amazon S3 Signature Version 2 authentication scheme.

2.1.2 Listing S3 Users

You can list all object storage users with the `ostor-s3-admin query-users` command. Information for each user can take one or more sequential rows in the table. Additional rows are used to lists S3 access key pairs associated with the user. If the user does not have any active key pairs, minus signs are shown in the corresponding table cells. For example:

```
# ostor-s3-admin query-users -V 0100000000000002
      S3 USER ID      S3 ACCESS KEY ID             S3 SECRET ACCESS KEY  S3 USER EMAIL
bf0b3b15eb7c9019  bf0b3b15eb7c9019I36Y                              ***  user2@abc.com
d866d9d114cc3d20  d866d9d114cc3d20G456                              ***  user1@abc.com
                  d866d9d114cc3d20D8EW                              ***
e86d1c19e616455                      -                                -  user3@abc.com
```

To output the list in XML, use the `-X` option; to output secret keys, use the `-a` option. For example:

```
# ostor-s3-admin query-users -V 0100000000000002 -a -X
<?xml version="1.0" encoding="UTF-8"?><QueryUsersResult><Users><User><Id>a49e12a226bd760f</Id><Ema
```

```
il>user@email.com</Email><Keys><OwnerId>0000000000000000</OwnerId><KeyPair><S3AccessKeyId>a49e12a2
26bd760fGHQ7</S3AccessKeyId><S3SecretAccessKey>HSDu2DA00JNGjnRcAhLKfhrvlymzOVdLPsCK2dcq</S3SecretA
ccessKey></KeyPair></Keys></User><User><Id>d7c53fc1f931661f</Id><Email>user@email.com</Email><Keys
><OwnerId>0000000000000000</OwnerId><KeyPair><S3AccessKeyId>d7c53fc1f931661fZLIV</S3AccessKeyId><S
3SecretAccessKey>JL7gt1OH873zR0Fzv8Oh9ZuA6JtCVnkgV7lET6ET</S3SecretAccessKey></KeyPair></Keys></Us
er></Users></QueryUsersResult>
```

2.1.3 Querying S3 User Information

To display information about the specified user, use the `ostor-s3-admin query-user-info` command. You need to specify either the user email (`-e`) or S3 ID (`-i`). For example:

```
# ostor-s3-admin query-user-info -e user@email.com -V 0100000000000002
Query user: user id=d866d9d114cc3d20, user email=user@email.com
Key pair[0]: access key id=d866d9d114cc3d20G456,
secret access key=5EAne6PLL1jxprouRqq8hmfONMfgrJcOwbowCoTt
Key pair[1]: access key id=d866d9d114cc3d20D8EW,
secret access key=83tTsNAuuRyoBBqhxMFqHAC60dhKHtTCCkQe54zu
```

2.1.4 Disabling S3 Users

You can disable a user with the `ostor-s3-admin disable-user` command. You need to specify either the user email (`-e`) or S3 ID (`-i`). For example:

```
# ostor-s3-admin disable-user -e user@email.com -V 0100000000000002
```

2.1.5 Deleting S3 Users

You can delete existing object storage users with the `ostor-s3-admin delete-user` command. Users who own any buckets cannot be deleted, so delete user's buckets first. You need to specify either the user email (`-e`) or S3 ID (`-i`). For example:

```
# ostor-s3-admin delete-user -i bf0b3b15eb7c9019 -V 0100000000000002
Deleted user: user id=bf0b3b15eb7c9019
```

2.1.6 Generating S3 User Access Key Pairs

You can generate a new access key pair for the specified user with the `ostor-s3-admin gen-access-key` command. The maximum of 2 active access key pairs are allowed per user (same as with the Amazon Web Services). You need to specify either the user email (`-e`) or S3 ID (`-i`). For example:

```
# ostor-s3-admin gen-access-key -e user@email.com -V 0100000000000002
Generate access key: user id=d866d9d114cc3d20, access key id=d866d9d114cc3d20D8EW,
secret access key=83tTsNAuuRyoBBqhxMFqHAC60dhKHtTCCkQe54zu
```

Note: It is recommended to periodically revoke old and generate new access key pairs.

2.1.7 Revoking S3 User Access Key Pairs

You can revoke the specified access key pair of the specified user with the `ostor-s3-admin revoke-access-key` command. You need to specify the access key in the key pair you want to delete as well as the user email or S3 ID. For example:

```
# ostor-s3-admin revoke-access-key -e user@email.com -k de86d1c19e616455YIPU -V 0100000000000002
Revoke access key: user id=de86d1c19e616455, access key id=de86d1c19e616455YIPU
```

2.2 Managing S3 User and Bucket Limits via CLI

This section describes limits you can define for users and buckets via the command-line interface. You can apply the limits according to specific options that can be a part of your service plan.

2.2.1 Setting Operations per Second for Users

You can limit operations rate with the `set-limits` command and the following parameters: `-e` specifying the email address, `-t ops` specifying the limit type, and `-L default=`, `get=`, `put=`, `list=`, or `delete=` specifying the limit key:

```
# ostor-s3-admin set-limits -e client@example.com -t ops -L get=3600
ops:default=0.00ops/s
ops:get=3600.00ops/s
ops:put=0.00ops/s
```

```
ops:list=0.00ops/s
ops:delete=0.00ops/s
bandwidth:out=0kbs/s
```

2.2.2 Setting Bandwidth per Second for Users

You can limit outgoing bandwidth of a response with the `set-limits` command and the following parameters: `-e` specifying the email address, `-t bandwidth` specifying the limit type, and `-L out=` specifying the limit key:

```
# ostor-s3-admin set-limits -e client@example.com -t bandwidth -L out=100
ops:default=0.00ops/s
ops:get=3600.00ops/s
ops:put=0.00ops/s
ops:list=0.00ops/s
ops:delete=0.00ops/s
bandwidth:out=100kbs/s
```

2.2.3 Querying User Limits

You can display the current limits with the `query-limits` command and parameter `-e` specifying the email address:

```
# ostor-s3-admin query-limits -e client@example.com
ops:default=0.00ops/s
ops:get=3600.00ops/s
ops:put=0.00ops/s
ops:list=0.00ops/s
ops:delete=0.00ops/s
bandwidth:out=100kbs/s
```

2.2.4 Deleting User Limits

You can delete the current limits with the `rm-limits` command and parameter `-e` specifying the email address:

```
# ostor-s3-admin rm-limits -e client@example.com
ops:default=0.00ops/s
ops:get=0.00ops/s
ops:put=0.00ops/s
ops:list=0.00ops/s
```

```
ops:delete=0.00ops/s
bandwidth:out=0kbs/s
```

2.2.5 Setting Operations per Second for Buckets

You can limit operations rate with the `set-limits` command and the following parameters: `-b` specifying the bucket name, `-t ops` specifying the limit type, and `-L default=`, `get=`, `put=`, `list=`, or `delete=` specifying the limit key:

```
# ostor-s3-admin set-limits -b example -t ops -L get=3600
ops:default=0.00ops/s
ops:get=3600.00ops/s
ops:put=0.00ops/s
ops:list=0.00ops/s
ops:delete=0.00ops/s
bandwidth:out=0kbs/s
```

2.2.6 Setting Bandwidth per Second for Buckets

You can limit outgoing bandwidth of a response with the `set-limits` command and the following parameters: `-b` specifying the bucket name, `-t bandwidth` specifying the limit type, and `-L out=` specifying the limit key:

```
# ostor-s3-admin set-limits -b example -t bandwidth -L out=100
ops:default=0.00ops/s
ops:get=3600.00ops/s
ops:put=0.00ops/s
ops:list=0.00ops/s
ops:delete=0.00ops/s
bandwidth:out=100kbs/s
```

2.2.7 Querying Bucket Limits

You can display the current limits with the `query-limits` command and parameter `-b` specifying the bucket name:

```
# ostor-s3-admin query-limits -b example
ops:default=0.00ops/s
ops:get=3600.00ops/s
ops:put=0.00ops/s
ops:list=0.00ops/s
ops:delete=0.00ops/s
```

```
bandwidth:out=100kbs/s
```

2.2.8 Deleting Bucket Limits

You can delete the current limits with the `rm-limits` command and parameter `-b` specifying the bucket name:

```
# ostor-s3-admin rm-limits -b example
ops:default=0.00ops/s
ops:get=0.00ops/s
ops:put=0.00ops/s
ops:list=0.00ops/s
ops:delete=0.00ops/s
bandwidth:out=0kbs/s
```

CHAPTER 3

Integration via REST API

This chapter explains ways to provision, enable, disable, and terminate S3 users as well as set user and bucket limits for billing purposes.

The provided examples are Bash commands with which you can send requests to S3 cluster's REST API via cURL and OpenSSL. Responses are in JSON format and can be processed further with tools like `json_pp` or `json_reformat`.

> **Note:** Replace `http://s3.example.com` in examples with your actual S3 gateway URL.

3.1 Requirements for Integration via REST API

Any operation or management request must be authenticated with a signed request via Signature Version 2 or 4 of the Amazon S3 protocol of the corresponding S3 system user. You can create system users on any storage node in the cluster with the `ostor-s3-admin create-user -S` command and parameter `-e` specifying the user email address:

```
# ostor-s3-admin create-user -S -e user@example.com
UserEmail:user@example.com
UserId:a14040e0b2ef8b28
KeyPair[0]:S3AccessKeyId:a14040e0b2ef8b28FZZ8
KeyPair[0]:S3SecretAccessKey:dbwTnQTW602aAAdq8DQVFzB6yrTCFTNiGB8C8RFA
Flags:system
```

With this user you will authenticate further REST API requests managing the S3 cluster. You can create multiple system accounts for different management operations.

3.1.1 Configuration

The examples will use cURL for authentication as well as `GET`, `PUT`, `POST`, and `DELETE` operations run in Bash. You can simplify signature creation if you create a sane environment first. Create a file `~/.s3_environment` replacing `s3_key` with your `S3AcessKeyId` and `s3_secret` with your `S3SecretAccessKey` and the following content:

```
# s3 login variables.
s3_key="a14040e0b2ef8b28FZZ8"
s3_secret="dbwTnQTW602aAAdq8DQVFzB6yrTCFTNiGB8C8RFA"

# Sign s3 requests and run curl.
function s3_curl() {

    # Parse command line.
    [ -z "${2}" ] && {
        echo "Usage: ${FUNCNAME[0]} <request_type> <s3_url>"
        return 1
    }

    # Prepare signature.
    s3_url="${2%/*}"
    s3_host="${s3_url##*://}"
    s3_query="${2##*/}"
    s3_date="$(date -R)"

    # Generate signature.
    s3_signature="$(echo -en "${1}\n\n\n${s3_date}\n/${s3_query%%&*}" |\
        openssl sha1 -hmac ${s3_secret} -binary | base64)"

    # Call.
    curl -H "Host: ${s3_host}" \
         -H "Accept: */*" \
         -H "Date: ${s3_date}" \
         -H "Authorization: AWS ${s3_key}:${s3_signature}" \
         -X "${1}" \
         "${s3_url}/${s3_query}"
}
```

You need to load the file into your default environment to make the `s3_curl` function available. Once the file is loaded, you can use it for all further S3 requests. Load the file with the following command:

```
# source ~/.s3_environment
```

3.1.2 Statistics

You need to have statistics collection enabled on your S3 gateway. The S3 gateway will save the statistics as regular storage objects. On each S3 storage node, create a file `/var/lib/ostor/local/gw.conf` with the following contents:

```
# Enable usage statistics collection.
S3_GW_COLLECT_STAT=1
```

Restart the S3 storage service to apply the configuration changes. Run the following command on all S3 storage nodes:

```
# systemctl restart ostor-agentd.service
```

3.2 Managing S3 Users and Listing Buckets via REST API

This section describes how to manage users via the REST API in a service provider scenario. New customers will sign up for the service during purchase in your online store and you will need to create users for them in the S3 cluster.

3.2.1 Creating S3 Users

You can create a user with the `ostor-users` service and parameter `emailAddress` specifying the user email address:

```
# s3_curl PUT "http://s3.example.com/?ostor-users&emailAddress=client@example.com"
{
    "UserEmail": "client@example.com",
    "UserId": "ca55631f9f3d59dc",
    "AWSAccessKeys": [
        {
            "AWSAccessKeyId": "ca55631f9f3d59dcDF4M",
            "AWSSecretAccessKey": "QCbj17BzeepyvUAdJeFNFYW9fCzbq0uFal6e5pGm"
        }
    ]
}
```

3.2.2 Listing S3 Users

You can list information about all users with the `ostor-users` service. Additional rows may list S3 access key pairs associated with the user. For example:

```
# s3_curl GET "http://s3.example.com/?ostor-users"
[
    {
        "UserEmail": "user@example.com",
        "UserId": "a14040e0b2ef8b28",
        "State": "enabled",
        "OwnerId": "0000000000000000"
    },
    {
        "UserEmail": "client@example.com",
        "UserId": "ca55631f9f3d59dc",
        "State": "enabled",
        "OwnerId": "0000000000000000"
    }
]
```

3.2.3 Querying S3 Users

You can display information and status of a user with the `ostor-users` service and parameter `emailAddress` specifying the user email address:

```
# s3_curl GET "http://s3.example.com/?ostor-users&emailAddress=client@example.com"
{
    "UserEmail": "client@example.com",
    "UserId": "ca55631f9f3d59dc",
    "State": "enabled",
    "OwnerId": "0000000000000000",
    "Flags": [

    ],
    "AWSAccessKeys": [
        {
            "AWSAccessKeyId": "ca55631f9f3d59dcDF4M",
            "AWSSecretAccessKey": "QCbj17BzeepyvUAdJeFNFYW9fCzbq0uFal6e5pGm"
        },
        {
            "AWSAccessKeyId": "ca55631f9f3d59dcZMDX",
            "AWSSecretAccessKey": "ffWvnOcNiH0jkQod4huv51BMYBuSWs4zRLFVwd4d"
        }
    ]
```

```
}
```

3.2.4 Disabling S3 Users

You can disable a user with the `ostor-users` service and parameter `emailAddress` specifying the user email address:

```
# s3_curl POST "http://s3.example.com/?ostor-users&emailAddress=client@example.com&disable"
```

3.2.5 Enabling S3 Users

You can enable a previously disabled user with the `ostor-users` service and parameter `emailAddress` specifying the user email address:

```
# s3_curl POST "http://s3.example.com/?ostor-users&emailAddress=client@example.com&enable"
```

3.2.6 Deleting S3 Users

You can delete existing users with the `ostor-users` service and parameter `emailAddress` specifying the user email address:

```
# s3_curl DELETE "http://s3.example.com/?ostor-users&emailAddress=client@example.com"
```

Users who own buckets cannot be removed until their buckets are deleted. You can get a list of user's buckets with the `ostor-buckets` service and parameter `emailAddress` specifying the user email address:

```
# s3_curl GET "http://s3.example.com/?ostor-buckets&emailAddress=client@example.com"
{
"Buckets": [
    {
        "size": {
            "current": 12288,
            "h_integral": 7360512,
            "hmax": 12288,
            "last_ts": 424241
        },
        "epoch": 0,
        "owner_id": "ba7eba06129464c5",
        "name": "data",
        "creation_date": "2018-05-25T17:12:00.000Z"
    }]
```

```
 {
```

You can delete the buckets by their names:

```
# s3_curl DELETE "http://s3.example.com/data"
```

3.2.7 Generating S3 Access Keys

You can generate a new or additional access key pair with the `ostor-users` service and the following parameters: `emailAddress` specifying the user email address, `genKey`:

```
# s3_curl POST "http://s3.example.com/?ostor-users&emailAddress=client@example.com&genKey"
{
    "UserEmail": "client@example.com",
    "UserId": "ca55631f9f3d59dc",
    "AWSAccessKeys": [
        {
            "AWSAccessKeyId": "ca55631f9f3d59dcZMDX",
            "AWSSecretAccessKey": "ffWvnOcNiH0jkQod4huv51BMYBuSWs4zRLFVwd4d"
        }
    ]
}
```

3.2.8 Revoking S3 Access Keys

You can revoke the specified access key pair of the specified user with the `ostor-users` service and the following parameters: `emailAddress` specifying the user email address, `revokeKey` specifying the access key in the key pair:

```
# s3_curl POST "http://s3.example.com/?ostor-users&emailAddress=client@example.com\
 &revokeKey=ca55631f9f3d59dcZMDX"
```

3.2.9 Listing User Buckets

You can list all buckets in S3 with the `ostor-buckets` service:

```
# s3_curl GET "http://s3.example.com/?ostor-buckets"
{
"Buckets": [
    {
        "size": {
            "current": 12288,
```

```
            "h_integral": 7360512,
            "hmax": 12288,
            "last_ts": 424241
        },
        "epoch": 0,
        "owner_id": "ba7eba06129464c5",
        "name": "bucket1",
        "creation_date": "2018-05-25T17:12:00.000Z"
    },
    {
        "size": {
            "current": 46700160,
            "h_integral": 28160196480,
            "hmax": 46700160,
            "last_ts": 424237
        },
        "epoch": 0,
        "owner_id": "ccbec013d9fd3918",
        "name": "bucket2",
        "creation_date": "2018-05-25T13:51:55.000Z"
    },
    {
        "size": {
            "current": 12288,
            "h_integral": 8036352,
            "hmax": 12288,
            "last_ts": 424186
        },
        "epoch": 0,
        "owner_id": "9d80d59edbe2862a",
        "name": "bucket3",
        "creation_date": "2018-05-23T10:30:49.000Z"
    }
]}
```


3.3 Managing S3 User and Bucket Limits via REST API

This section describes limits you can define for users and buckets via REST API. You can apply the limits according to specific options that can be a part of your service plan.

3.3.1 Setting Operations per Second for Users

You can limit operations rate with the `ostor-limits` service and the following parameters: `emailAddress` specifying the email address, `ops` specifying the limit type, and `default=`, `get=`, `put=`, `list=`, or `delete=` specifying the limit value:

```
# s3_curl PUT "http://s3.example.com/?ostor-limits&emailAddress=client@example.com&limit-type=ops\
 &limit-resource=get&limit-value=3600"
```

3.3.2 Setting Bandwidth per Second for Users

You can limit outgoing bandwidth of a response with the `ostor-limits` service and the following parameters: `emailAddress` specifying the email address, `bandwidth` specifying the limit type, and `out=` specifying the limit value:

```
# s3_curl PUT "http://s3.example.com/?ostor-limits&emailAddress=client@example.com\
 &limit-type=bandwidth&limit-resource=out&limit-value=100"
```

3.3.3 Querying User Limits

You can display the current limits with the `ostor-limits` service and parameter `emailAddress` specifying the email address:

```
# s3_curl GET "http://s3.example.com/?ostor-limits&emailAddress=client@example.com"
{
    "ops:default": "0.00",
    "ops:get": "3600.00",
    "ops:put": "0.00",
    "ops:list": "0.00",
    "ops:delete": "0.00",
    "bandwidth:out": "100"
}
```

3.3.4 Deleting User Limits

You can delete the current limits with the `ostor-limits` service and parameter `emailAddress` specifying the email address:

```
# s3_curl DELETE "http://s3.example.com/?ostor-limits&emailAddress=client@example.com"
```

3.3.5 Setting Operations per Second for Buckets

You can limit operations rate with the `ostor-limits` service and the following parameters: `bucket` specifying the bucket name, `ops` specifying the limit type, and `default=`, `get=`, `put=`, `list=`, or `delete=` specifying the limit value:

```
# s3_curl PUT "http://s3.example.com/?ostor-limits&bucket=client&limit-type=ops\
 &limit-resource=get&limit-value=3600"
```

3.3.6 Setting Bandwidth per Second for Buckets

You can limit outgoing bandwidth of a response with the `ostor-limits` service and the following parameters: `bucket` specifying the bucket name, `bandwidth` specifying the limit type, and `out=` specifying the limit value:

```
# s3_curl PUT "http://s3.example.com/?ostor-limits&bucket=client&limit-type=bandwidth\
 &limit-resource=out&limit-value=100"
```

3.3.7 Querying Bucket Limits

You can display the current limits with the `ostor-limits` service and parameter `bucket` specifying the bucket name:

```
# s3_curl GET "http://s3.example.com/?ostor-limits&bucket=client"
{
    "ops:default": "0.00",
    "ops:get": "3600.00",
    "ops:put": "0.00",
    "ops:list": "0.00",
    "ops:delete": "0.00",
    "bandwidth:out": "100"
}
```

3.3.8 Deleting Bucket Limits

You can delete the current limits with the `ostor-limits` service and parameter `bucket` specifying the bucket name:

```
# s3_curl DELETE "http://s3.example.com/?ostor-limits&bucket=client"
```

3.4 Obtaining Usage Statistics via REST API

Thi section describes how to obtain usage statistics via REST API for billing or other purposes.

Note:	Delete statistics objects after collecting the required data.

3.4.1 Listing Statistics Objects

You can list all available statistics objects with the `ostor-usage` service and no parameters. The output only contains objects that have not been deleted. For example:

```
# s3_curl GET "http://s3.example.com/?ostor-usage"
{
    "nr_items": 7,
    "truncated": false,
    "items": [
        "s3-usage-8000000000000065-2017-02-01T16:31:54.000Z-1800",
        "s3-usage-8000000000000067-2017-02-01T16:30:51.000Z-1800",
        "s3-usage-8000000000000068-2017-02-01T16:27:25.000Z-1800",
        "s3-usage-8000000000000069-2017-02-01T16:27:24.000Z-1800",
        "s3-usage-8000000000000069-2017-02-01T16:31:07.000Z-1800",
        "s3-usage-800000000000006a-2017-02-01T16:27:24.000Z-1800",
        "s3-usage-800000000000006a-2017-02-01T16:31:08.000Z-1800"
    ]
}
```

3.4.2 Querying Statistics Objects

You can display usage statistics with the `ostor-usage` service and parameter `obj` specifying the statistics object. The output includes the accessed buckets, user ID, and counters. For example:

```
# s3_curl GET "http://s3.example.com/?ostor-usage\
 &obj=s3-usage-8000000000000065-2017-02-01T16:31:54.000Z-1800"
{
    "fmt_version": 1,
    "service_id": 8000000000000065,
    "start_ts": 1485966714,
    "period": 1390,
    "nr_items": 1,
    "items": [
        {
```

```
            "key": {
                "bucket": "client",
                "epoch": 98309,
                "user_id": "b81d6c5f895a8c86",
                "tag": ""
            },
            "counters": {
                "ops": {
                    "put": 1,
                    "get": 3,
                    "list": 0,
                    "other": 0
                },
                "net_io": {
                    "uploaded": 41258,
                    "downloaded": 45511311
                }
            }
        }
    ]
}
```

3.4.3 Deleting Statistics Objects

You can delete existing statistics objects with the `ostor-usage` service and parameter `obj` specifying the statistics object:

```
# s3_curl DELETE "http://s3.example.com/?ostor-usage\
 &obj=s3-usage-8000000000000065-2017-02-01T16:31:54.000Z-1800"
```

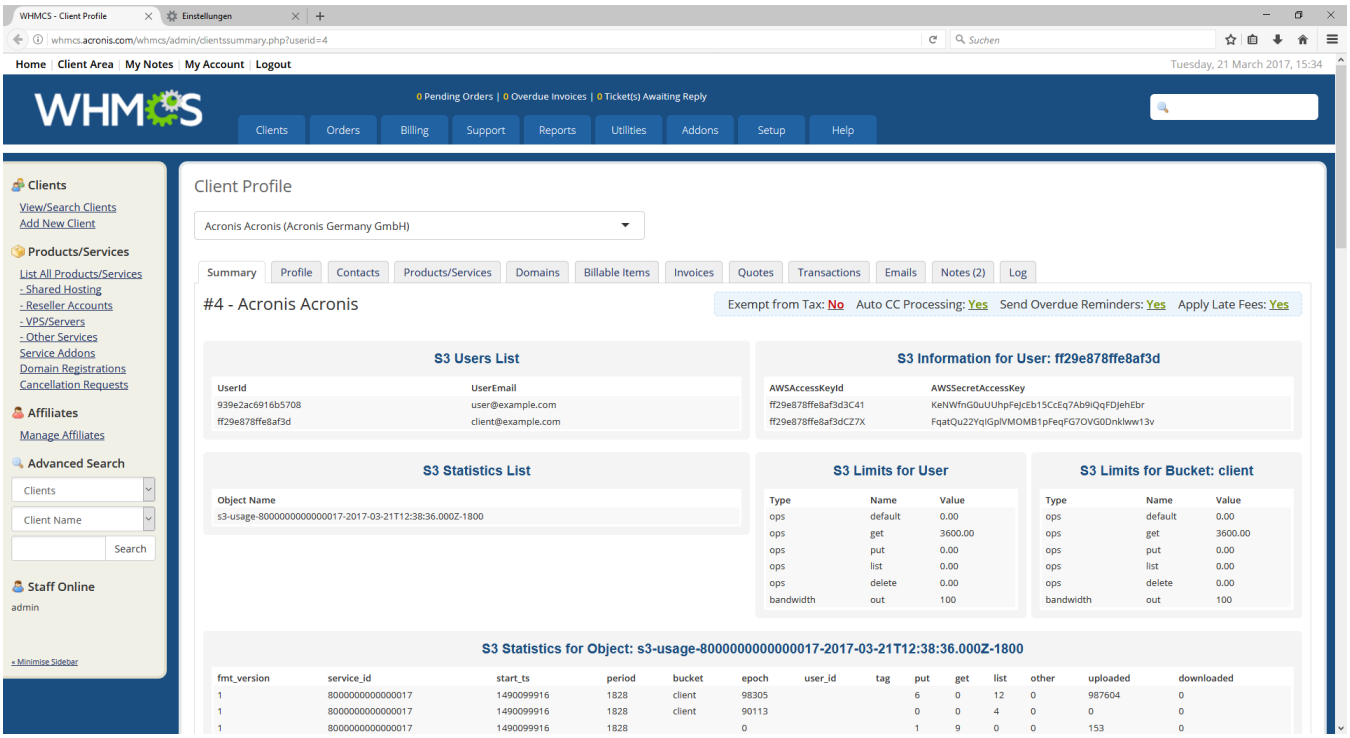
CHAPTER 4

Integration with WHMCS

This chapter explains ways to provision, enable, disable, and terminate S3 users as well as set user and bucket limits for billing purposes.

The provided examples are PHP scripts with which you can send requests to S3 cluster's REST API via cURL and OpenSSL.

Note: Replace `http://s3.example.com` in examples with your actual S3 gateway URL and `http://whmcs.example.com` with you actual WHMCS portal URL.



4.1 Requirements

Any operation or management request must be authenticated with a signed request via Signature Version 2 or 4 of the Amazon S3 protocol of the corresponding S3 system user. You can create system users on any storage node in the cluster with the `ostor-s3-admin create-user -S` command and parameter `-e` specifying the user email address:

```
# ostor-s3-admin create-user -S -e user@example.com
UserEmail:user@example.com
UserId:a14040e0b2ef8b28
KeyPair[0]:S3AccessKeyId:a14040e0b2ef8b28FZZ8
KeyPair[0]:S3SecretAccessKey:dbwTnQTW602aAAdq8DQVFzB6yrTCFTNiGB8C8RFA
Flags:system
```

With this user you will authenticate further REST API requests managing the S3 cluster. You can create multiple system accounts for different management operations.

4.1.1 Configuration

In addition, you need to create Acronis Storage directories to modify the default functionality.

Change to the document root directory of your WHMCS server (e.g., `/srv/http`) and create the following directories in it:

- `whmcs/includes/staas_scripts`,
- `whmcs/admin/staas_scripts`.

Change to the directory `whmcs/includes/staas_scripts`.

The first file you need to create includes the S3 configuration. Create a configuration file `S3_getConfig.php` with the following contents, replacing variables as follows:

- `s3_key` with your `S3AcessKeyId`,
- `s3_secret` with your `S3SecretAccessKey`,
- `s3_gateway` with your configured S3 gateway address, and
- `whmcs_username` with your WHMCS admin username.

```
<?php
```

```
// Return array with default configuration.
if (!function_exists('S3_getConfig')) {
    function S3_getConfig() {

        // s3 login.
        $vars['s3_key'] = "939e2ac6916b57082P9O";
        $vars['s3_secret'] = "tVYF3kZD9zcTtl6q6QDTHaZKM2nuq4xVcl8ikJpd";

        // s3 gateway.
        $vars['s3_gateway'] = "http://s3.example.com";

        // whmcs login.
        $vars['whmcs_username'] = "admin";

        // Return config array.
        return $vars;
    }
}

?>
```

4.1.2 Includes

Shared functions required by API operations are provided in a number of standalone PHP include files. The first file returns the client information (e.g., email address) which further S3 API user management requests need for various operations. Create a file `S3_getClient.php` with the following contentss:

```
<?php

// API request to get whmcs client information.
if (!function_exists('S3_getClient')) {
    function S3_getClient($userid, $whmcs_username) {

        // Get client details for user email.
        $command = 'GetClientsDetails';
        $data = array(
            'clientid' => $userid,
        );
        $results = localAPI($command, $data, $whmcs_username);

        // Return client information.
     return $results;
    }
}

?>
```

The next file adds notes to the client in WHMCS with the S3 access key pairs whenever a new user or access key pair is created. Create a file `S3_addClientNote.php` with the following contentss:

```
<?php

// API request to add note to client in whmcs.
if (!function_exists('S3_addClientNote')) {
    function S3_addClientNote(
        $userid,
        $whmcs_username,
        $s3_client_userid,
        $s3_client_key,
        $s3_client_secret
    ) {

        // Add note only for non-empty users.
        if (!empty($s3_client_userid)) {

            // Add note with the s3 access key and s3 secret.
            $command = 'AddClientNote';
            $data = array(
                'userid' => $userid,
                'notes' =>
                    "UserId: " . $s3_client_userid . "\n" .
                    "AWSAccessKeyId: " . $s3_client_key . "\n" .
                    "AWSSecretAccessKey: " . $s3_client_secret,
            );
            localAPI($command, $data, $whmcs_username);
        }
    }
}

?>
```

The next file removes notes from the client in WHMCS with the S3 access key pairs whenever a user or access key pair is removed. Create a file `S3_delClientNote.php` with the following contents:

```
<?php

// whmcs database access.
use WHMCS\Database\Capsule;

// API request to remove note from client in whmcs.
if (!function_exists('S3_delClientNote')) {
    function S3_delClientNote(
        $userid,
        $whmcs_username,
        $s3_client_userid,
        $s3_client_key
```

```
    ) {

        // Delete notes in database.
        $db = Capsule::connection()->getPdo();
        $db->exec('
            DELETE FROM
              tblnotes
            WHERE
              userid = ' . $userid . '
            AND
              note LIKE "%' . $s3_client_userid . '%"
            AND
              note LIKE "%' . $s3_client_key . '%"'
        );
    }
}

?>
```

The last file is the cURL library for sending `GET`, `PUT`, `POST`, and `DELETE` requests. Create a file `S3_requestCurl.php` with the following contents:

```
<?php

// API request to s3 gateway.
if (!function_exists('S3_requestCurl')) {
    function S3_requestCurl($s3_key, $s3_secret, $s3_gateway, $s3_query, $method) {

        // Prepare signature.
        $s3_host  = parse_url($s3_gateway, PHP_URL_HOST);
        $s3_date  = date(DATE_RFC2822);

        // Generate signature.
        $s3_signature = hash_hmac('sha1', $method . "\n\n\n" . $s3_date . "\n" .
            current(explode('&', $s3_query)), $s3_secret, true);
        $s3_signature = base64_encode($s3_signature);

        // Curl init.
        $s3_curl = curl_init($s3_gateway . $s3_query);

        // Curl options.
        switch ($method) {
            case "PUT":
                curl_setopt($s3_curl, CURLOPT_PUT, 1);
                break;
            case "POST":
                curl_setopt($s3_curl, CURLOPT_POST, 1);
                break;
            case "DELETE":
```

```
                curl_setopt($s3_curl, CURLOPT_CUSTOMREQUEST, "DELETE");
                break;
        }
        curl_setopt($s3_curl, CURLOPT_RETURNTRANSFER, true);
        curl_setopt($s3_curl, CURLOPT_URL, $s3_gateway . $s3_query);
        curl_setopt($s3_curl, CURLOPT_HTTPHEADER, array(
            'Host: ' . $s3_host,
            'Date: ' . $s3_date,
            'Authorization: AWS ' . $s3_key . ':' . $s3_signature,
            'Content-Type:',
            'Expect:',
        ));

        // Call.
        $response = curl_exec($s3_curl);
        $response = json_decode($response, true);

        // Curl deinit.
        curl_close($s3_curl);

        // Return response.
        return $response;
    }
}

?>
```

4.1.3 Hooks

Hooks allow you to execute custom code when certain events occur in WHMCS. You will need to add S3-related action links to the admin page in WHMCS.

Change to the directory `whmcs/includes/hooks` and create a file `S3_adminAreaClientSummaryActionLinks.php` with the following contents:

```
<?php

// Modify other actions admin page.
function S3_adminAreaClientSummaryActionLinks($vars) {

    // Create additional links.
    $result[] = '<b>S3 - User Management</b>';
    $result[] = '<a href="staas_scripts/S3_createUser.php?userid=' .
        $vars['userid'] . '"><img src="http://logo.acronis.com/ogimage.png"
            width="16" height="16" border="0" align="absmiddle" /> Create User</a>';
    $result[] = '<a href="staas_scripts/S3_deleteUser.php?userid=' .
```

```
        $vars['userid'] . '"><img src="http://logo.acronis.com/ogimage.png"
            width="16" height="16" border="0" align="absmiddle" /> Delete User</a>';
    $result[] = '<a href="staas_scripts/S3_enableUser.php?userid=' .
        $vars['userid'] . '"><img src="http://logo.acronis.com/ogimage.png"
            width="16" height="16" border="0" align="absmiddle" /> Enable User</a>';
    $result[] = '<a href="staas_scripts/S3_disableUser.php?userid=' .
        $vars['userid'] . '"><img src="http://logo.acronis.com/ogimage.png"
            width="16" height="16" border="0" align="absmiddle" /> Disable User</a>';
    $result[] = '<a href="staas_scripts/S3_generateAccessKey.php?userid=' .
        $vars['userid'] . '"><img src="http://logo.acronis.com/ogimage.png"
            width="16" height="16" border="0" align="absmiddle" /> Generate Access Key</a>';
    $result[] = '<a href="staas_scripts/S3_revokeAccessKey.php?userid=' .
        $vars['userid'] . '"><img src="http://logo.acronis.com/ogimage.png"
            width="16" height="16" border="0" align="absmiddle" /> Revoke Access Key</a>';
    $result[] = '<a href="staas_scripts/S3_queryUser.php?userid=' .
        $vars['userid'] . '"><img src="http://logo.acronis.com/ogimage.png"
            width="16" height="16" border="0" align="absmiddle" /> Query User (on/off)</a>';
    $result[] = '<a href="staas_scripts/S3_listUsers.php">
        <img src="http://logo.acronis.com/ogimage.png"
            width="16" height="16" border="0" align="absmiddle" /> List Users (on/off)</a>';
    $result[] = ' ';
    $result[] = '<b>S3 - User Limits Management</b>';
    $result[] = '
        <form>
            <input name="userid" type="hidden" value="' . $vars['userid'] . '">
            <input name="ops-value" size="4">
            <select name="ops-name">
                <option>default</option>
                <option>get</option>
                <option>put</option>
                <option>list</option>
                <option>delete</option>
            </select> ops/s
            <br />
            <input name="bandwidth-value" size="4">
            <select name="bandwidth-name">
                <option>out</option>
            </select> bandwidth/s
            <br />
            <button type="submit"
                formaction="staas_scripts/S3_setLimitsForUser.php">Set</button>
            <button type="submit"
                formaction="staas_scripts/S3_getLimitsForUser.php">Get</button>
            <button type="submit"
                formaction="staas_scripts/S3_deleteLimitsForUser.php">Delete</button>
        </form>
    ';
    $result[] = ' ';
    $result[] = '<b>S3 - Bucket Limits Management</b>';
```

```
    $result[] = '
        <form>
            <input name="userid" type="hidden" value="' . $vars['userid'] . '">
            <input name="ops-value" size="4">
            <select name="ops-name">
                <option>default</option>
                <option>get</option>
                <option>put</option>
                <option>list</option>
                <option>delete</option>
            </select> ops/s
            <br />
            <input name="bandwidth-value" size="4">
            <select name="bandwidth-name">
                <option>out</option>
            </select> bandwidth/s
            <br />
            <input name="bucket" size="4"> bucket name
            <br />
            <button type="submit"
                formaction="staas_scripts/S3_setLimitsForBucket.php">Set</button>
            <button type="submit"
                formaction="staas_scripts/S3_getLimitsForBucket.php">Get</button>
            <button type="submit"
                formaction="staas_scripts/S3_deleteLimitsForBucket.php">Delete</button>
        </form>
    ';
    $result[] = ' ';
    $result[] = '<b>S3 - Usage Statistics</b>';
    $result[] = '
        <a href="staas_scripts/S3_listStatsObjects.php">
            <img src="http://logo.acronis.com/ogimage.png"
                width="16" height="16" border="0" align="absmiddle" />
                    List Statistics Objects (on/off)
        </a>
        <p>
            <form>
                <input name="object" size="15"> object name
                <br />
                <button type="submit"
                    formaction="staas_scripts/S3_getStatsForObject.php">Get</button>
                <button type="submit"
                    formaction="staas_scripts/S3_deleteStatsForObject.php">Delete</button>
            </form>
        </p>
    ';
    $result[] = ' ';

    // Return links.
```

```
    return $result;
}

// Modify admin area.
add_hook('AdminAreaClientSummaryActionLinks', 1, "S3_adminAreaClientSummaryActionLinks");
?>
```

The last file extends the admin summary page and displays S3 user information as well as user and bucket limits if the corresponding links are clicked. Create a file `S3_adminAreaClientSummaryPage.php` with the following contents:

```
<?php

// Modify admin client summary to show S3 information.
function S3_adminAreaClientSummaryPage($vars) {

    // Sane default.
    $result = '
    <div class="row client-summary-panels">
    ';

    // Show users.
    if ($_SESSION['s3_list_users'] == 1) {

        // Table header.
        $result = $result . '
        <div class="col-lg-6 col-sm-12">
            <div class="clientssummarybox">
                <div class="title">
                    S3 Users List
                </div>
                <table class="clientssummarystats" cellspacing="0" cellpadding="2">
                    <tr>
                        <td><b>UserId</b></td>
                        <td><b>UserEmail</b></td>
                    </tr>
        ';

        // One row per access key pair.
        foreach ($_SESSION['s3_list'] as $s3_row) {
            $result = $result . '
                    <tr class="altrow">
                        <td>' . $s3_row['UserId'] . '</td>
                        <td>' . $s3_row['UserEmail'] . '</td>
                    </tr>
            ';
        }

        // Table footer.
```

```
        $result = $result . '
                </table>
            </div>
        </div>
        ';
    }

    // Show user.
    if ($_SESSION['s3_query_user'] == 1) {

        // Table header.
        $result = $result . '
        <div class="col-lg-6 col-sm-12">
            <div class="clientssummarybox">
                <div class="title">
                    S3 Information for User: ' . $_SESSION['s3_userid'] . '
                </div>
                <table class="clientssummarystats" cellspacing="0" cellpadding="2">
                    <tr>
                        <td><b>AWSAccessKeyId</b></td>
                        <td><b>AWSSecretAccessKey</b></td>
                    </tr>
        ';

        // One row per access key pair.
        foreach ($_SESSION['s3_aws_access_keys'] as $s3_row) {
            $result = $result . '
                    <tr class="altrow">
                        <td>' . $s3_row['AWSAccessKeyId'] . '</td>
                        <td>' . $s3_row['AWSSecretAccessKey'] . '</td>
                    </tr>
            ';
        }

        // Table footer.
        $result = $result . '
                </table>
            </div>
        </div>
        ';
    }

    // Table footer and next header.
    $result = $result . '
    </div>
    <div class="row client-summary-panels">
    ';

    // Show statistics list.
```

```
    if ($_SESSION['s3_stat_objects'] == 1) {

        // Table header.
        $result = $result . '
        <div class="col-lg-6 col-sm-12">
            <div class="clientssummarybox">
                <div class="title">
                    S3 Statistics List
                </div>
                <table class="clientssummarystats" cellspacing="0" cellpadding="2">
                    <tr>
                        <td><b>Object Name</b></td>
                    </tr>
        ';

        // One row per access key pair.
        foreach ($_SESSION['s3_stat']['items'] as $s3_object) {
            $result = $result . '
                    <tr class="altrow">
                        <td>' . $s3_object . '</td>
                    </tr>
            ';
        }

        // Table footer.
        $result = $result . '
                </table>
            </div>
        </div>
        ';
    }

    // Show limits for user.
    if (!empty($_SESSION['s3_limits_user'])) {

        // Table header.
        $result = $result . '
        <div class="col-lg-3 col-sm-6">
            <div class="clientssummarybox">
                <div class="title">
                    S3 Limits for User
                </div>
                <table class="clientssummarystats" cellspacing="0" cellpadding="2">
                    <tr>
                        <td><b>Type</b></td>
                        <td><b>Name</b></td>
                        <td><b>Value</b></td>
                    </tr>
        ';
```

```
        // One row per access key pair.
        foreach ($_SESSION['s3_limits_user'] as $s3_limits => $s3_value) {
            list($s3_type, $s3_limit) = explode(":", $s3_limits);
            $result = $result . '
                    <tr class="altrow">
                        <td>' . $s3_type . '</td>
                        <td>' . $s3_limit . '</td>
                        <td>' . $s3_value . '</td>
                    </tr>
            ';
        }

        // Table footer.
        $result = $result . '
                </table>
            </div>
        </div>
        ';
    }

    // Show limits for bucket.
    if (!empty($_SESSION['s3_limits_bucket'])) {

        // Table header.
        $result = $result . '
        <div class="col-lg-3 col-sm-6">
            <div class="clientssummarybox">
                <div class="title">
                    S3 Limits for Bucket: ' . $_SESSION['s3_bucket'] . '
                </div>
                <table class="clientssummarystats" cellspacing="0" cellpadding="2">
                    <tr>
                        <td><b>Type</b></td>
                        <td><b>Name</b></td>
                        <td><b>Value</b></td>
                    </tr>
        ';

        // One row per access key pair.
        foreach ($_SESSION['s3_limits_bucket'] as $s3_limits => $s3_value) {
            list($s3_type, $s3_limit) = explode(":", $s3_limits);
            $result = $result . '
                    <tr class="altrow">
                        <td>' . $s3_type . '</td>
                        <td>' . $s3_limit . '</td>
                        <td>' . $s3_value . '</td>
                    </tr>
            ';
```

```
        }

        // Table footer.
        $result = $result . '
                </table>
            </div>
        </div>
        ';
    }

    // Table footer and next header.
    $result = $result . '
    </div>
    <div class="row client-summary-panels">
    ';

    // Show statistics for object.
    if (!empty($_SESSION['s3_object_statistic'])) {

        // Table header.
        $result = $result . '
        <div class="col-lg-12 col-sm-24">
            <div class="clientssummarybox">
                <div class="title">
                    S3 Statistics for Object: ' . $_SESSION['s3_object'] . '
                </div>
                <table class="clientssummarystats" cellspacing="0" cellpadding="2">
                    <tr>
                        <td><b>fmt_version</b></td>
                        <td><b>service_id</b></td>
                        <td><b>start_ts</b></td>
                        <td><b>period</b></td>
                        <td><b>bucket</b></td>
                        <td><b>epoch</b></td>
                        <td><b>user_id</b></td>
                        <td><b>tag</b></td>
                        <td><b>put</b></td>
                        <td><b>get</b></td>
                        <td><b>list</b></td>
                        <td><b>other</b></td>
                        <td><b>uploaded</b></td>
                        <td><b>downloaded</b></td>
                    </tr>
        ';

        // One row per access key pair.
        foreach ($_SESSION['s3_object_statistic']['items'] as $s3_object) {
            $result = $result . '
                    <tr class="altrow">
```

```
                        <td>' . $_SESSION['s3_object_statistic']['fmt_version']. '</td>
                        <td>' . $_SESSION['s3_object_statistic']['service_id']. '</td>
                        <td>' . $_SESSION['s3_object_statistic']['start_ts']. '</td>
                        <td>' . $_SESSION['s3_object_statistic']['period']. '</td>
                        <td>' . $s3_object['key']['bucket'] . '</td>
                        <td>' . $s3_object['key']['epoch'] . '</td>
                        <td>' . $s3_object['key']['user'] . '</td>
                        <td>' . $s3_object['key']['tag'] . '</td>
                        <td>' . $s3_object['counters']['ops']['put'] . '</td>
                        <td>' . $s3_object['counters']['ops']['get'] . '</td>
                        <td>' . $s3_object['counters']['ops']['list'] . '</td>
                        <td>' . $s3_object['counters']['ops']['other'] . '</td>
                        <td>' . $s3_object['counters']['net_io']['uploaded'] . '</td>
                        <td>' . $s3_object['counters']['net_io']['downloaded'] . '</td>
                    </tr>
            ';
        }

        // Table footer.
        $result = $result . '
                </table>
            </div>
        </div>
        ';
    }

    // Table footer.
    $result = $result . '
    </div>
    ';

    // Return table.
    return $result;
}

// Modify admin area.
add_hook('AdminAreaClientSummaryPage', 1, "S3_adminAreaClientSummaryPage");
?>
```

4.1.4 Statistics

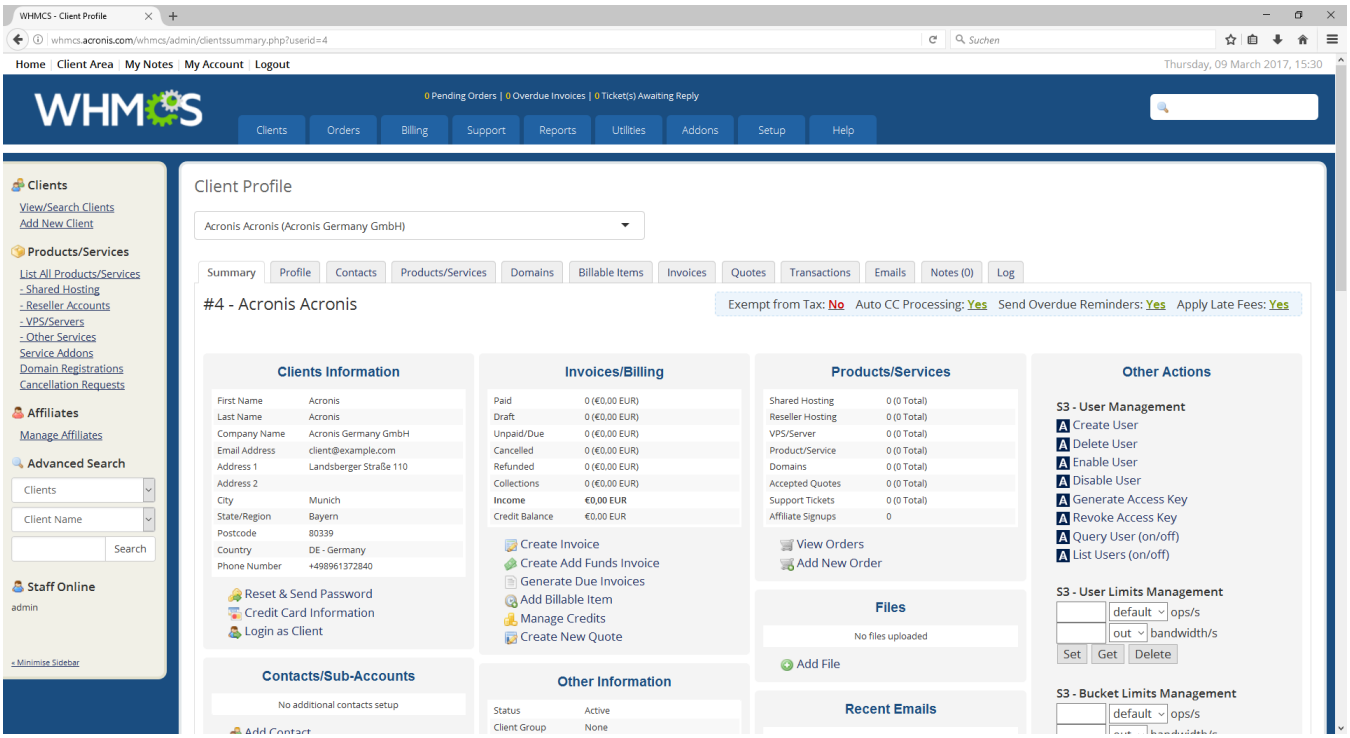
You need to have statistics collection enabled on your S3 gateway. The S3 gateway will save the statistics as regular storage objects. On each S3 storage node, create a file `/var/lib/ostor/local/gw.conf` with the following contents:

```
# Enable usage statistics collection.
S3_GW_COLLECT_STAT=1
```

Restart the S3 storage service to apply the configuration changes. Run the following command on all S3 storage nodes:

```
# systemctl restart ostor-agentd.service
```

Now you can login to WHMCS. Additional links and S3 management options will be shown in the **Client Profile** section.



4.2 Managing S3 Users in WHMCS

This section describes how to manage users in WHMCS in a service provider scenario. New customers will sign up for the service during purchase in your online store and you will need to create users for them in the S3 cluster.

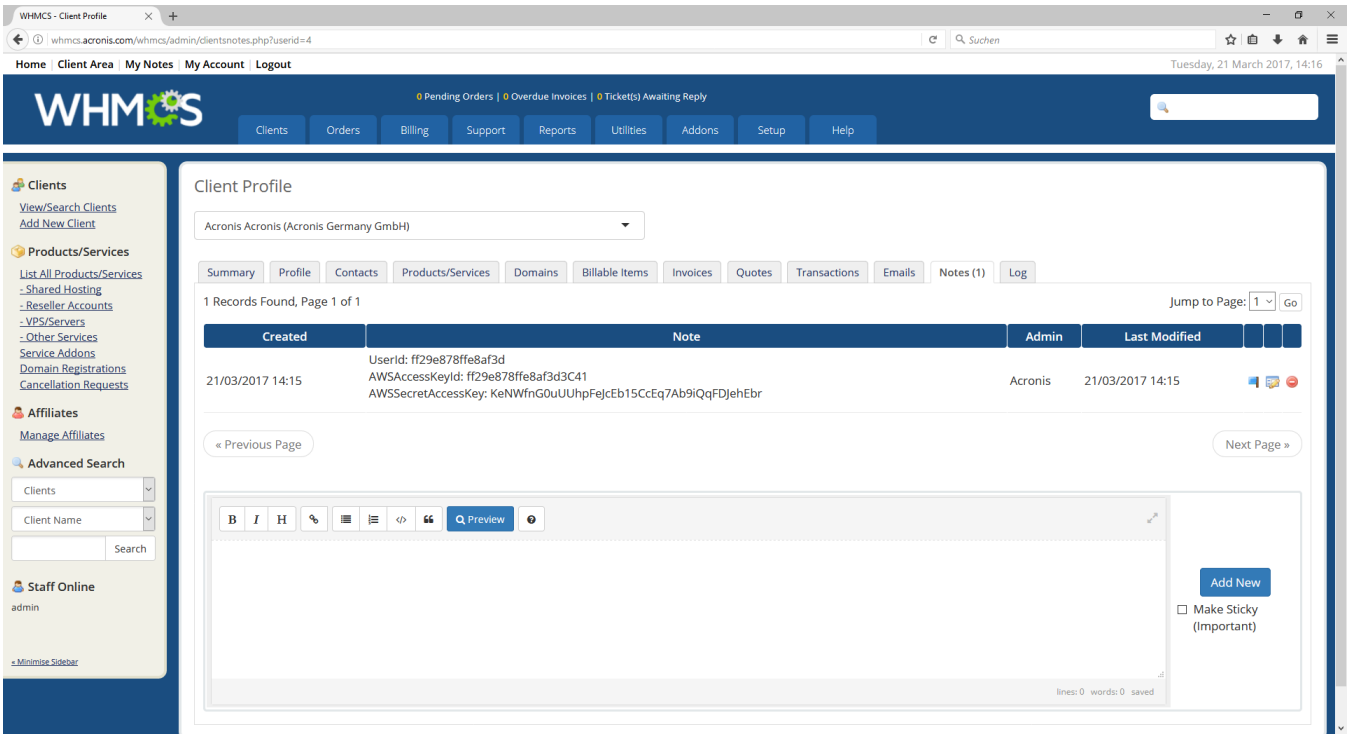
Create all files mentioned further in the directory `whmcs/admin/staas_scripts`.

4.2.1 Creating S3 Users

You can create a user with the `ostor-users` service and parameter `emailAddress` specifying the user email address. WHMCS creates the user in S3 cluster when you click **Create User**. Create a file `S3_createUser.php` with the following contents:

```
<?php

// Load configuration and libraries.
require('../../includes/staas_scripts/S3_addClientNote.php');
require('../../includes/staas_scripts/S3_getClient.php');
require('../../includes/staas_scripts/S3_getConfig.php');
require('../../includes/staas_scripts/S3_requestCurl.php');
require('../../init.php');

// Create s3 user.
function S3_createUser($userid) {

    // Load configuration.
    $s3_config = s3_getConfig();

    // Get whmcs user email.
    $s3_whmcs = S3_getClient($userid, $s3_config['whmcs_username']);

    // Create s3 user.
    $s3_client = S3_requestCurl(
        $s3_config['s3_key'],
        $s3_config['s3_secret'],
        $s3_config['s3_gateway'],
        "/?ostor-users&emailAddress=" . $s3_whmcs['email'],
        "PUT"
    );

    // Add note with the s3 access key and s3 secret.
    S3_addClientNote(
        $s3_whmcs['userid'],
        $s3_config['whmcs_username'],
        $s3_client['UserId'],
        $s3_client['AWSAccessKeys']['0']['AWSAccessKeyId'],
        $s3_client['AWSAccessKeys']['0']['AWSSecretAccessKey']
    );

    // Redirect back.
    header('Location: ' . $_SERVER['HTTP_REFERER']);
}

// Call function.
S3_createUser($_GET['userid']);
```

```
?>
```



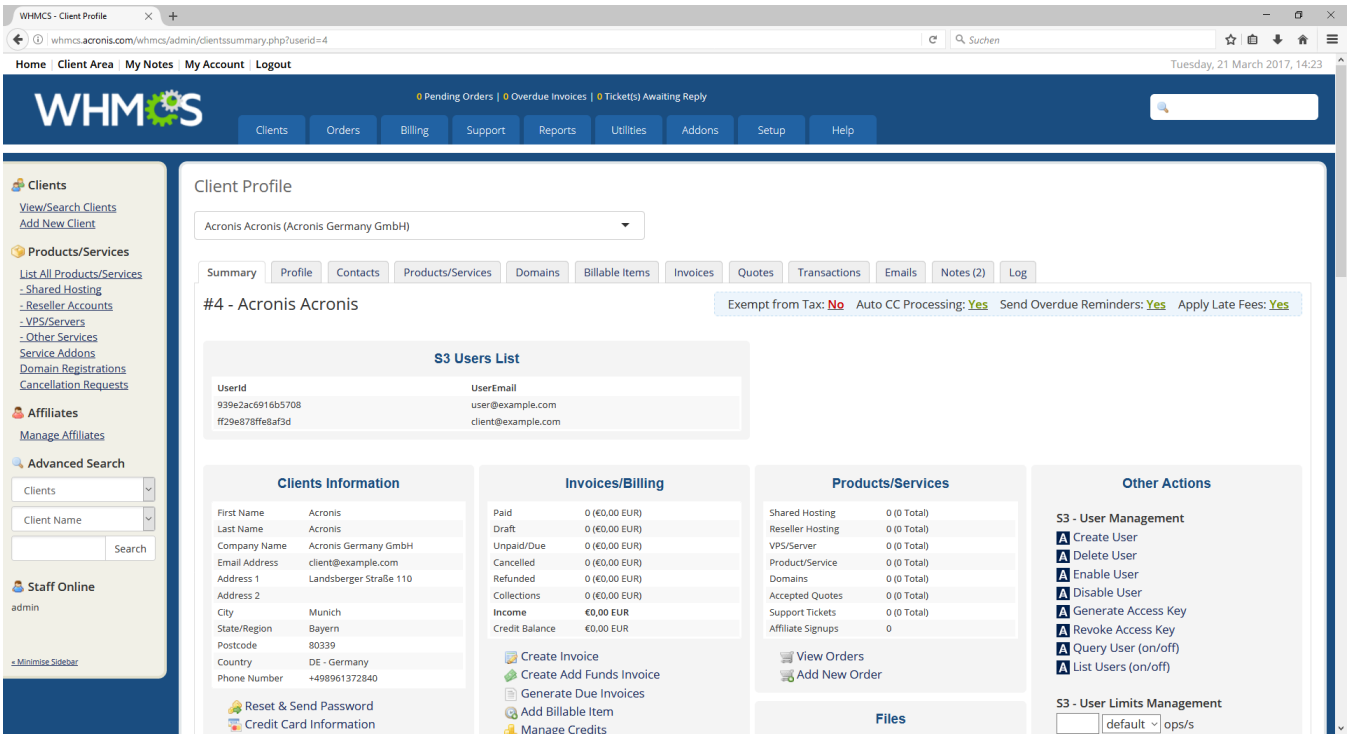
4.2.2 Listing S3 Users

You can list information about all users with the `ostor-users` service. Additional rows may list S3 access key pairs associated with the user. WHMCS lists the users information fetched from S3 cluster when you click **List Users (on/off)**. Create a file `S3_listUsers.php` with the following contents:

```
<?php

// Load configuration and libraries.
require('../../includes/staas_scripts/S3_getConfig.php');
require('../../includes/staas_scripts/S3_requestCurl.php');
require('../../init.php');

// List s3 users.
function S3_listUsers() {

    // Hide now.
    if ($_SESSION['s3_list_users'] == 1) {

        // Hide.
```

```
        $_SESSION['s3_list_users'] = 0;

        // Redirect back.
        header('Location: ' . $_SERVER['HTTP_REFERER']);

     // Return immediately.
        return;
    }

    // Load configuration.
    $s3_config = s3_getConfig();

    // Get s3 users.
    $s3_client = S3_requestCurl(
        $s3_config['s3_key'],
        $s3_config['s3_secret'],
        $s3_config['s3_gateway'],
        "/?ostor-users",
        "GET"
    );

    // Store s3 result.
    $_SESSION['s3_list_users'] = 1;
    $_SESSION['s3_list'] = $s3_client;

    // Redirect back.
    header('Location: ' . $_SERVER['HTTP_REFERER']);
}

// Call function.
S3_listUsers();

?>
```

4.2. Managing S3 Users in WHMCS



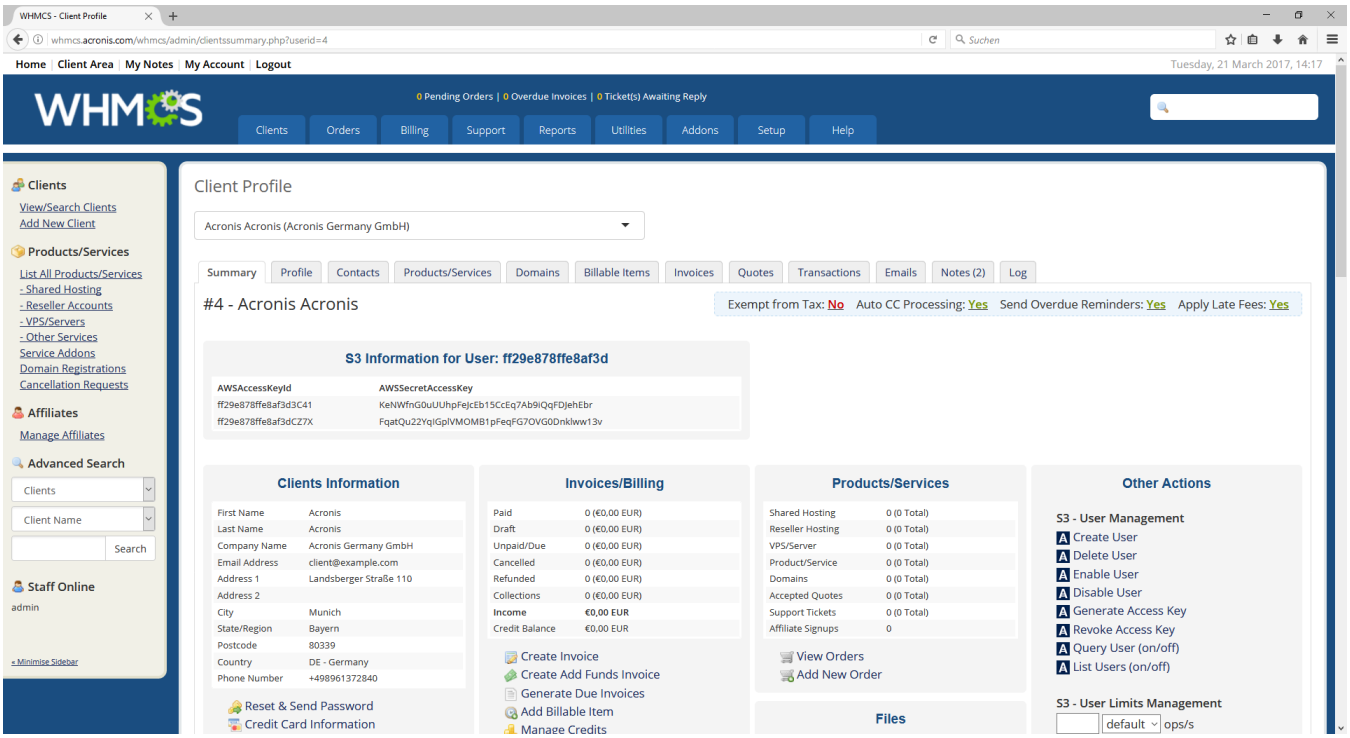
4.2.3 Querying S3 Users

You can display information and status of a user with the `ostor-users` service and parameter `emailAddress` specifying the user email address. WHMCS displays the user information fetched from S3 cluster when you click **Query User (on/off)**. Create a file `S3_queryUser.php` with the following contents:

```
<?php

// Load configuration and libraries.
require('../../includes/staas_scripts/S3_getClient.php');
require('../../includes/staas_scripts/S3_getConfig.php');
require('../../includes/staas_scripts/S3_requestCurl.php');
require('../../init.php');

// Query s3 user.
function S3_queryUser($userid) {

    // Hide now.
    if ($_SESSION['s3_query_user'] == 1) {

        // Hide.
        $_SESSION['s3_query_user'] = 0;

        // Redirect back.
```

```
        header('Location: ' . $_SERVER['HTTP_REFERER']);

     // Return immediately.
        return;
    }

    // Load configuration.
    $s3_config = s3_getConfig();

    // Get whmcs user email.
    $s3_whmcs = S3_getClient($userid, $s3_config['whmcs_username']);

    // Get s3 user id.
    $s3_client = S3_requestCurl(
        $s3_config['s3_key'],
        $s3_config['s3_secret'],
        $s3_config['s3_gateway'],
        "/?ostor-users&emailAddress=" . $s3_whmcs['email'],
        "GET"
    );

    // Store s3 result.
    $_SESSION['s3_query_user'] = 1;
    $_SESSION['s3_userid'] = $s3_client['UserId'];
    $_SESSION['s3_aws_access_keys'] = $s3_client['AWSAccessKeys'];

    // Redirect back.
    header('Location: ' . $_SERVER['HTTP_REFERER']);
}

// Call function.
S3_queryUser($_GET['userid']);

?>
```

4.2. Managing S3 Users in WHMCS



4.2.4 Disabling S3 Users

You can disable users with the `ostor-users` service and parameter `emailAddress` specifying the user email address. WHMCS disables read and write access to S3 cluster when you click **Disable User**. Create a file `S3_disableUser.php` with the following contents:

```
<?php

// Load configuration and libraries.
require('../../includes/staas_scripts/S3_getClient.php');
require('../../includes/staas_scripts/S3_getConfig.php');
require('../../includes/staas_scripts/S3_requestCurl.php');
require('../../init.php');

// Disable user.
function S3_disableUser($userid) {

    // Load configuration.
    $s3_config = s3_getConfig();

    // Get whmcs user email.
    $s3_whmcs = S3_getClient($userid, $s3_config['whmcs_username']);

    // Disable user.
```

```
    $s3_client = S3_requestCurl(
        $s3_config['s3_key'],
        $s3_config['s3_secret'],
        $s3_config['s3_gateway'],
        "/?ostor-users&emailAddress=" . $s3_whmcs['email'] . "&disable",
        "POST"
    );

    // Redirect back.
    header('Location: ' . $_SERVER['HTTP_REFERER']);
}

// Call function.
S3_disableUser($_GET['userid']);

?>
```

4.2.5 Enabling S3 Users

You can enable a previously disabled user with the `ostor-users` service and parameter `emailAddress` specifying the user email address. WHMCS enables read and write access to S3 cluster for user when you click **Enable User**. Create a file `S3_enableUser.php` with the following contents:

```
<?php

// Load configuration and libraries.
require('../../includes/staas_scripts/S3_getClient.php');
require('../../includes/staas_scripts/S3_getConfig.php');
require('../../includes/staas_scripts/S3_requestCurl.php');
require('../../init.php');

// Enable user.
function S3_enableUser($userid) {

    // Load configuration.
    $s3_config = s3_getConfig();

    // Get whmcs user email.
    $s3_whmcs = S3_getClient($userid, $s3_config['whmcs_username']);

    // Enable user.
    $s3_client = S3_requestCurl(
        $s3_config['s3_key'],
        $s3_config['s3_secret'],
        $s3_config['s3_gateway'],
        "/?ostor-users&emailAddress=" . $s3_whmcs['email'] . "&enable",
```

```
        "POST"
    );

    // Redirect back.
    header('Location: ' . $_SERVER['HTTP_REFERER']);
}

// Call function.
S3_enableUser($_GET['userid']);

?>
```

4.2.6 Deleting S3 Users

You can delete users with the `ostor-users` service and parameter `emailAddress` specifying the user email address. WHMCS removes the user from S3 cluster when you click **Delete User**. Create a file `S3_deleteUser.php` with the following contents:

```
<?php

// Load configuration and libraries.
require('../../includes/staas_scripts/S3_delClientNote.php');
require('../../includes/staas_scripts/S3_getClient.php');
require('../../includes/staas_scripts/S3_getConfig.php');
require('../../includes/staas_scripts/S3_requestCurl.php');
require('../../init.php');

// Delete s3 user.
function S3_deleteUser($userid) {

    // Load configuration.
    $s3_config = s3_getConfig();

    // Get whmcs user email.
    $s3_whmcs = S3_getClient($userid, $s3_config['whmcs_username']);

    // Get s3 user id.
    $s3_client = S3_requestCurl(
        $s3_config['s3_key'],
        $s3_config['s3_secret'],
        $s3_config['s3_gateway'],
        "/?ostor-users&emailAddress=" . $s3_whmcs['email'],
        "GET"
    );

    // Delete s3 user.
```

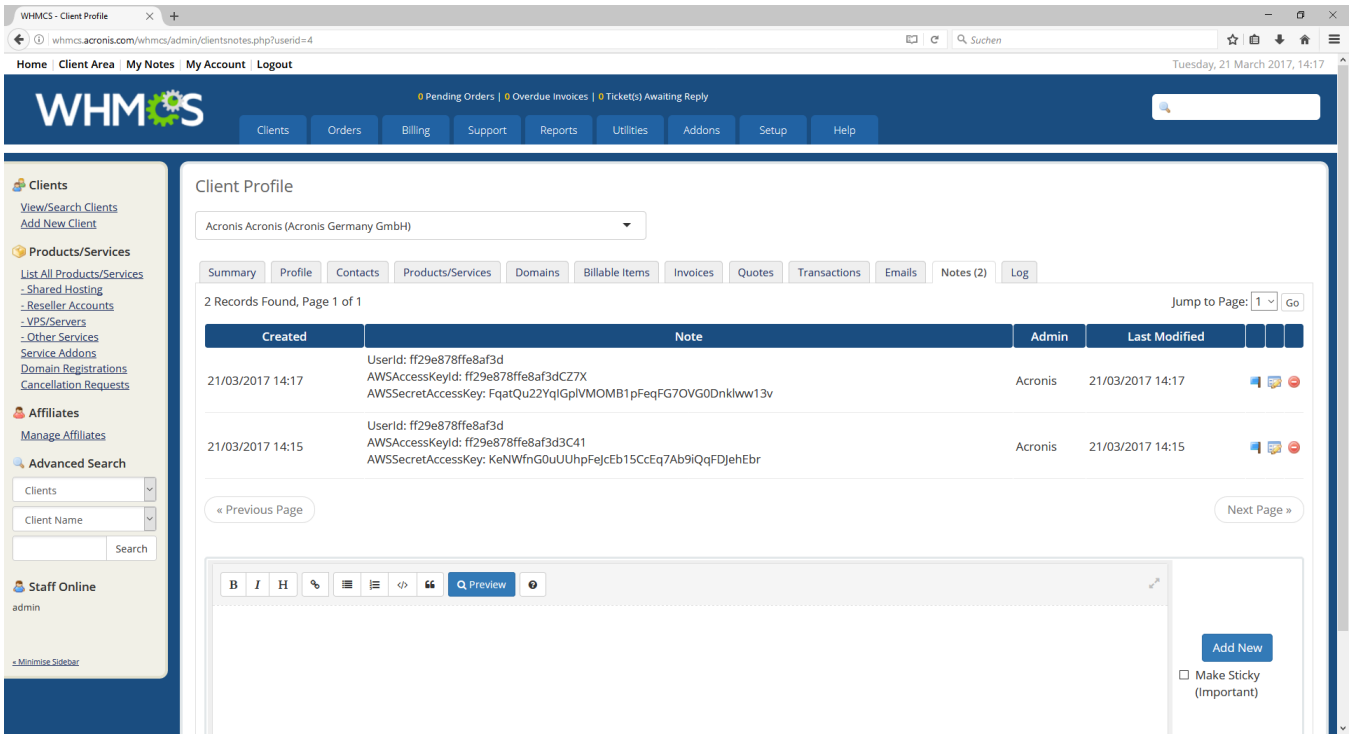
```
    S3_requestCurl(
        $s3_config['s3_key'],
        $s3_config['s3_secret'],
        $s3_config['s3_gateway'],
        "/?ostor-users&emailAddress=" . $s3_whmcs['email'],
        "DELETE"
    );

    // Delete note with the s3 access key and s3 secret.
    S3_delClientNote(
        $s3_whmcs['userid'],
        $s3_config['whmcs_username'],
        $s3_client['UserId'],
        ""
    );

    // Redirect back.
    header('Location: ' . $_SERVER['HTTP_REFERER']);
}

// Call function.
S3_deleteUser($_GET['userid']);

?>
```

4.2.7 Generating S3 Access Keys

You can generate a new or additional access key pair with the `ostor-users` service and the following parameters: `emailAddress` specifying the user email address, `genKey`. WHMCS generates a new key pair when you click **Generate Access Key**. Create a file `S3_generateAccessKey.php` with the following contents:

```
<?php

// Load configuration and libraries.
require('../../includes/staas_scripts/S3_addClientNote.php');
require('../../includes/staas_scripts/S3_getClient.php');
require('../../includes/staas_scripts/S3_getConfig.php');
require('../../includes/staas_scripts/S3_requestCurl.php');
require('../../init.php');

// Generate s3 access key pair.
function S3_generateAccessKey($userid) {

    // Load configuration.
    $s3_config = s3_getConfig();
```

```
    // Get whmcs user email.
    $s3_whmcs = S3_getClient($userid, $s3_config['whmcs_username']);

    // Generate s3 key pair.
    $s3_client = S3_requestCurl(
        $s3_config['s3_key'],
        $s3_config['s3_secret'],
        $s3_config['s3_gateway'],
        "/?ostor-users&emailAddress=" . $s3_whmcs['email'] . "&genKey",
        "POST"
    );

    // Add note with the s3 access key and s3 secret.
    S3_addClientNote(
        $s3_whmcs['userid'],
        $s3_config['whmcs_username'],
        $s3_client['UserId'],
        $s3_client['AWSAccessKeys']['0']['AWSAccessKeyId'],
        $s3_client['AWSAccessKeys']['0']['AWSSecretAccessKey']
    );

    // Redirect back.
    header('Location: ' . $_SERVER['HTTP_REFERER']);
}

// Call function.
S3_generateAccessKey($_GET['userid']);

?>
```



4.2.8 Revoking S3 Access Keys

You can revoke the specified access key pair of the specified user with the `ostor-users` service and the following parameters: `emailAddress` specifying the user email address, `revokeKey` specifying the access key in the key pair. WHMCS removes the key pair when you click **Revoke Access Key**. Create a file `S3_revokeAccessKey.php` with the following contents:

```
<?php

// Load configuration and libraries.
require('../../includes/staas_scripts/S3_delClientNote.php');
require('../../includes/staas_scripts/S3_getClient.php');
require('../../includes/staas_scripts/S3_getConfig.php');
require('../../includes/staas_scripts/S3_requestCurl.php');
require('../../init.php');

// Revoke s3 access key pair.
function S3_revokeAccessKey($userid) {

    // Load configuration.
    $s3_config = s3_getConfig();

    // Get whmcs user email.
```

```
    $s3_whmcs = S3_getClient($userid, $s3_config['whmcs_username']);

    // Get first s3 access key.
    $s3_client = S3_requestCurl(
        $s3_config['s3_key'],
        $s3_config['s3_secret'],
        $s3_config['s3_gateway'],
        "/?ostor-users&emailAddress=" . $s3_whmcs['email'],
        "GET"
    );

    // Revoke s3 access key.
    S3_requestCurl(
        $s3_config['s3_key'],
        $s3_config['s3_secret'],
        $s3_config['s3_gateway'],
            "/?ostor-users&emailAddress=" . $s3_whmcs['email'] .
            "&revokeKey=" . $s3_client['AWSAccessKeys']['0']['AWSAccessKeyId'],
        "POST"
    );

    // Delete note with the s3 access key and s3 secret.
    S3_delClientNote(
        $s3_whmcs['userid'],
        $s3_config['whmcs_username'],
        $s3_client['UserId'],
        $s3_client['AWSAccessKeys']['0']['AWSAccessKeyId']
    );

    // Redirect back.
    header('Location: ' . $_SERVER['HTTP_REFERER']);
}

// Call function.
S3_revokeAccessKey($_GET['userid']);

?>
```

4.3 Managing S3 User and Bucket Limits in WHMCS

This section describes limits you can define for users and buckets in WHMCS. You can apply the limits according to specific options that can be a part of your service plan.

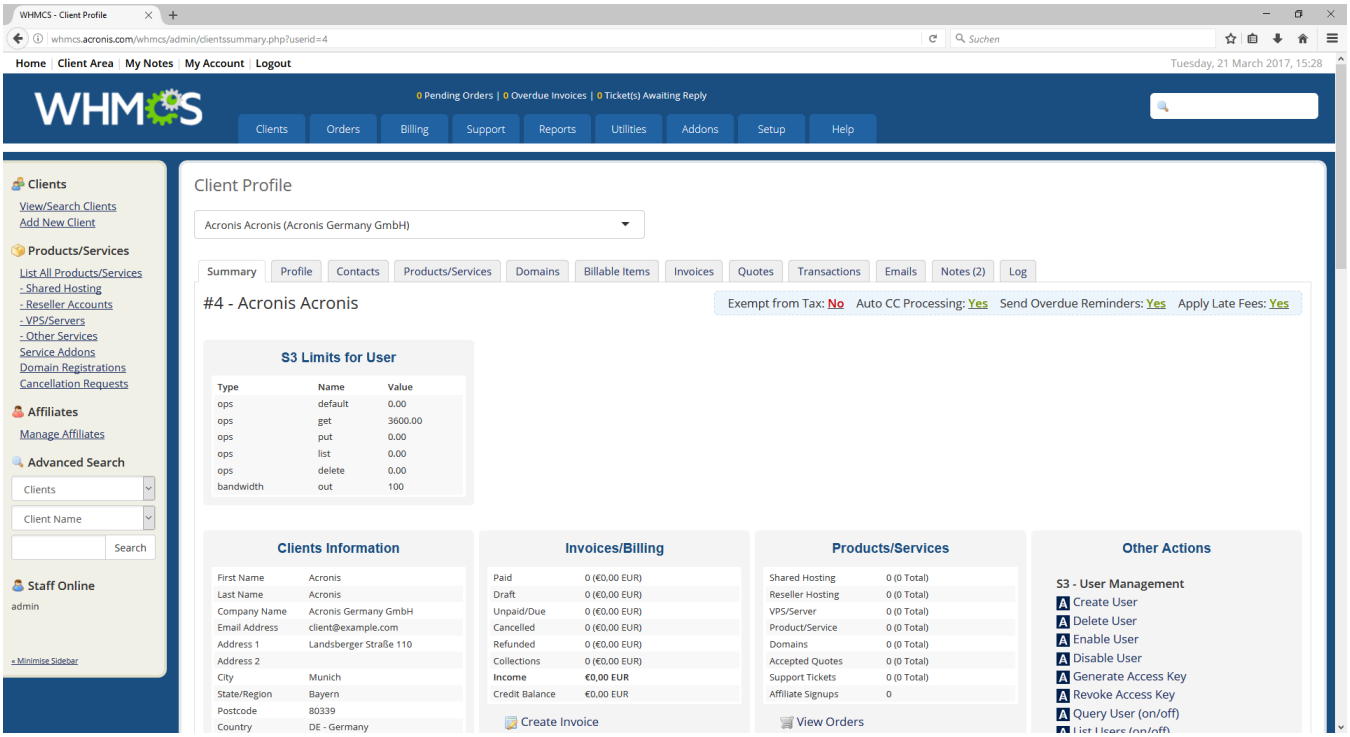
4.3.1 Setting User Limits

You can limit operations rate with the `ostor-limits` service and the following parameters: `emailAddress` specifying the email address, `default=`, `get=`, `put=`, `list=`, or `delete=` specifying the limit value.

Similarly, you can limit outgoing bandwidth of a response with the following parameters: `emailAddress` specifying the email address, `out=` specifying the limit value. WHMCS configures user limits in an S3 cluster when you click the **Set** button. Create a file `S3_setLimitsForUser.php` with the following contents:

```
<?php

// Load configuration and libraries.
require('../../includes/staas_scripts/S3_getClient.php');
require('../../includes/staas_scripts/S3_getConfig.php');
require('../../includes/staas_scripts/S3_requestCurl.php');
require('../../init.php');

// Set s3 user limits.
function S3_setLimitsForUser($vars) {

    // Load configuration.
    $s3_config = s3_getConfig();

    // Get whmcs user email.
    $s3_whmcs = S3_getClient($vars['userid'], $s3_config['whmcs_username']);

    // Set only if value specified.
    if (!empty($vars['ops-value'])) {

        // Set s3 bucket limits (ops).
        S3_requestCurl(
            $s3_config['s3_key'],
            $s3_config['s3_secret'],
            $s3_config['s3_gateway'],
                "/?ostor-limits&emailAddress=" . $s3_whmcs['email'] .
                "&limit-type=ops&limit-resource=" . $vars['ops-name'] .
                    '&limit-value=' . $vars['ops-value'],
            "PUT"
        );
    }

    // Set only if value specified.
    if (!empty($vars['bandwidth-value'])) {

        // Set s3 bucket limits (bandwidth).
        S3_requestCurl(
            $s3_config['s3_key'],
```

```
            $s3_config['s3_secret'],
            $s3_config['s3_gateway'],
                "/?ostor-limits&emailAddress=" . $s3_whmcs['email'] .
                "&limit-type=bandwidth&limit-resource=" . $vars['bandwidth-name'] .
                    '&limit-value=' . $vars['bandwidth-value'],
            "PUT"
        );
    }

    // Redirect back.
    header('Location: ' . $_SERVER['HTTP_REFERER']);
}

// Call function.
S3_setLimitsForUser($_GET);

?>
```



4.3.2 Querying User Limits

You can display the current limits with the `ostor-limits` service and parameter `emailAddress` specifying the email address. WHMCS displays the user limits in S3 cluster when you click the **Get** button. Create a file `S3_getLimitsForUser.php` with the following contents:

```
<?php

// Load configuration and libraries.
require('../../includes/staas_scripts/S3_getClient.php');
require('../../includes/staas_scripts/S3_getConfig.php');
require('../../includes/staas_scripts/S3_requestCurl.php');
require('../../init.php');

// Get s3 user limits.
function S3_getLimitsForUser($userid) {

    // Load configuration.
    $s3_config = s3_getConfig();

    // Get whmcs user email.
    $s3_whmcs = S3_getClient($userid, $s3_config['whmcs_username']);

    // Get s3 user limits.
    $s3_client = S3_requestCurl(
        $s3_config['s3_key'],
        $s3_config['s3_secret'],
        $s3_config['s3_gateway'],
        "/?ostor-limits&emailAddress=" . $s3_whmcs['email'],
        "GET"
    );

    // Store s3 result.
    $_SESSION['s3_limits_user'] = $s3_client;

    // Redirect back.
    header('Location: ' . $_SERVER['HTTP_REFERER']);
}

// Call function.
S3_getLimitsForUser($_GET['userid']);

?>
```

4.3.3 Deleting User Limits

You can delete the current limits with the `ostor-limits` service and parameter `emailAddress` specifying the email address. WHMCS removes the user limits from S3 cluster when you click the **Delete** button. Create a file `S3_deleteLimitsForUser.php` with the following contents:

```
<?php
```

```
// Load configuration and libraries.
require('../../includes/staas_scripts/S3_getClient.php');
require('../../includes/staas_scripts/S3_getConfig.php');
require('../../includes/staas_scripts/S3_requestCurl.php');
require('../../init.php');

// Delete s3 user limits.
function S3_getLimitsForUser($userid) {

    // Load configuration.
    $s3_config = s3_getConfig();

    // Get whmcs user email.
    $s3_whmcs = S3_getClient($userid, $s3_config['whmcs_username']);

    // Delete s3 user limits.
    S3_requestCurl(
        $s3_config['s3_key'],
        $s3_config['s3_secret'],
        $s3_config['s3_gateway'],
        "/?ostor-limits&emailAddress=" . $s3_whmcs['email'],
        "DELETE"
    );

    // Clear array.
    $_SESSION['s3_limits_user'] = null;

    // Redirect back.
    header('Location: ' . $_SERVER['HTTP_REFERER']);
}

// Call function.
S3_getLimitsForUser($_GET['userid']);

?>
```

4.3.4 Setting Buckets Limits

You can limit operations rate with the `ostor-limits` service and the following parameters: `bucket` specifying the bucket name, `default=`, `get=`, `put=`, `list=`, `delete=` specifying the limit value.

Similarly, you can limit outgoing bandwidth of a response with the `ostor-limits` service and the following parameters: `bucket` specifying the bucket name, `out=` specifying the limit value. WHMCS configures the bucket limits in S3 cluster when you click the **Set** button. Create a file `S3_setLimitsForBucket.php` with the following contents:

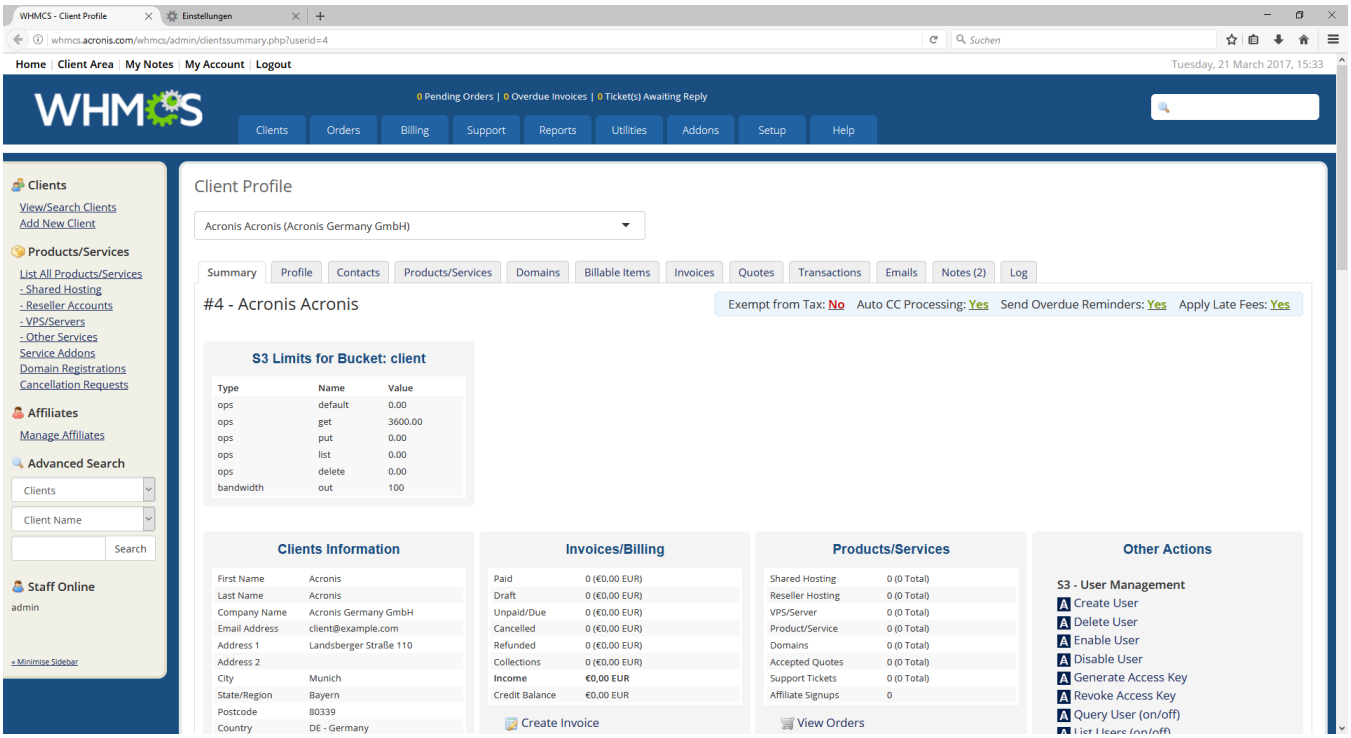
```
<?php

// Load configuration and libraries.
require('../../includes/staas_scripts/S3_getConfig.php');
require('../../includes/staas_scripts/S3_requestCurl.php');
require('../../init.php');

// Set s3 bucket limits.
function S3_setLimitsForBucket($vars) {

    // Load configuration.
    $s3_config = s3_getConfig();

    // Set only if value specified.
    if (!empty($vars['ops-value'])) {

        // Set s3 bucket limits (ops).
        S3_requestCurl(
            $s3_config['s3_key'],
            $s3_config['s3_secret'],
            $s3_config['s3_gateway'],
                "/?ostor-limits&bucket=" . $vars['bucket'] .
                "&limit-type=ops&limit-resource=" . $vars['ops-name'] .
                    '&limit-value=' . $vars['ops-value'],
            "PUT"
        );
    }

    // Set only if value specified.
    if (!empty($vars['bandwidth-value'])) {

        // Set s3 bucket limits (bandwidth).
        S3_requestCurl(
            $s3_config['s3_key'],
            $s3_config['s3_secret'],
            $s3_config['s3_gateway'],
                "/?ostor-limits&bucket=" . $vars['bucket'] .
                "&limit-type=bandwidth&limit-resource=" . $vars['bandwidth-name'] .
                    '&limit-value=' . $vars['bandwidth-value'],
            "PUT"
        );
    }

    // Redirect back.
    header('Location: ' . $_SERVER['HTTP_REFERER']);
}

// Call function.
S3_setLimitsForBucket($_GET);
```

```
?>
```

4.3.5 Querying Bucket Limits

You can display the current limits with the `ostor-limits` service and parameter `bucket` specifying the bucket name. WHMCS displays the bucket limits in S3 cluster when you click the **Get** button. Create a file `S3_getLimitsForBucket.php` with the following contents:

```
<?php

// Load configuration and libraries.
require('../../includes/staas_scripts/S3_getConfig.php');
require('../../includes/staas_scripts/S3_requestCurl.php');
require('../../init.php');

// Get s3 bucket limits.
function S3_getLimitsForBucket($bucket) {

    // Load configuration.
    $s3_config = s3_getConfig();

    // Get s3 user limits.
    $s3_client = S3_requestCurl(
        $s3_config['s3_key'],
        $s3_config['s3_secret'],
        $s3_config['s3_gateway'],
        "/?ostor-limits&bucket=" . $bucket,
        "GET"
    );

    // Store s3 result.
    $_SESSION['s3_limits_bucket'] = $s3_client;
    $_SESSION['s3_bucket'] = $bucket;

    // Redirect back.
    header('Location: ' . $_SERVER['HTTP_REFERER']);
}

// Call function.
S3_getLimitsForBucket($_GET['bucket']);

?>
```



4.3.6 Deleting Bucket Limits

You can delete the current limits with the `ostor-limits` service and parameter `bucket` specifying the bucket name. WHMCS removes the bucket limits from S3 cluster when you click the **Delete** button. Create a file `S3_deleteLimitsForBucket.php` with the following contents:

```
<?php

// Load configuration and libraries.
require('../../includes/staas_scripts/S3_getConfig.php');
require('../../includes/staas_scripts/S3_requestCurl.php');
require('../../init.php');

// Delete s3 bucket limits.
function S3_deleteLimitsForBucket($bucket) {

    // Load configuration.
    $s3_config = s3_getConfig();

    // Delete s3 bucket limits.
    S3_requestCurl(
        $s3_config['s3_key'],
        $s3_config['s3_secret'],
        $s3_config['s3_gateway'],
```

```
        "/?ostor-limits&bucket=" . $bucket,
        "DELETE"
    );

    // Clear array.
    $_SESSION['s3_limits_bucket'] = null;

    // Redirect back.
    header('Location: ' . $_SERVER['HTTP_REFERER']);
}

// Call function.
S3_deleteLimitsForBucket($_GET['bucket']);

?>
```

4.4 Obtaining Usage Statistics in WHMCS

Thi section describes how to obtain usage statistics via in WHMCS for billing or other purposes.

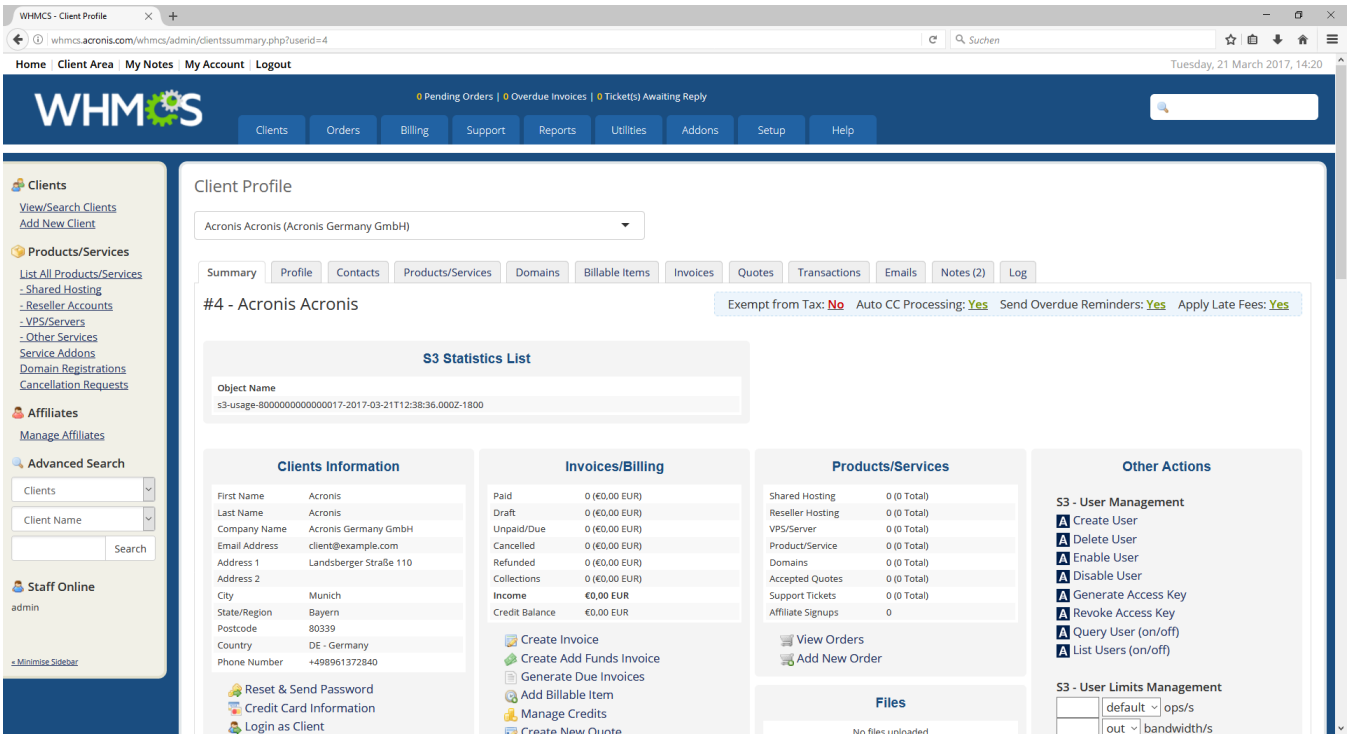
> **Note:** Delete statistics objects after collecting the required data.

4.4.1 Listing Statistics Objects

You can list all available statistics objects with the `ostor-usage` service and no parameters. IThe output only contains objects that have not been deleted. WHMCS lists the available statistics objects from S3 cluster when you click **List statistics objects (on/off)**. Create a file `S3_listStatsObjects.php` with the following contents:

```
<?php

// Load configuration and libraries.
require('../../includes/staas_scripts/S3_getConfig.php');
require('../../includes/staas_scripts/S3_requestCurl.php');
require('../../init.php');

// List s3 statistics objects.
function S3_listStatsObjects() {

    // Hide now.
    if ($_SESSION['s3_stat_objects'] == 1) {

        // Hide.
```

```
        $_SESSION['s3_stat_objects'] = 0;

        // Redirect back.
        header('Location: ' . $_SERVER['HTTP_REFERER']);

     // Return immediately.
        return;
    }

    // Load configuration.
    $s3_config = s3_getConfig();

    // Get s3 statistics objects.
    $s3_client = S3_requestCurl(
        $s3_config['s3_key'],
        $s3_config['s3_secret'],
        $s3_config['s3_gateway'],
        "/?ostor-usage",
        "GET"
    );

    // Store s3 result.
    $_SESSION['s3_stat_objects'] = 1;
    $_SESSION['s3_stat'] = $s3_client;

    // Redirect back.
    header('Location: ' . $_SERVER['HTTP_REFERER']);
}

// Call function.
S3_listStatsObjects();

?>
```

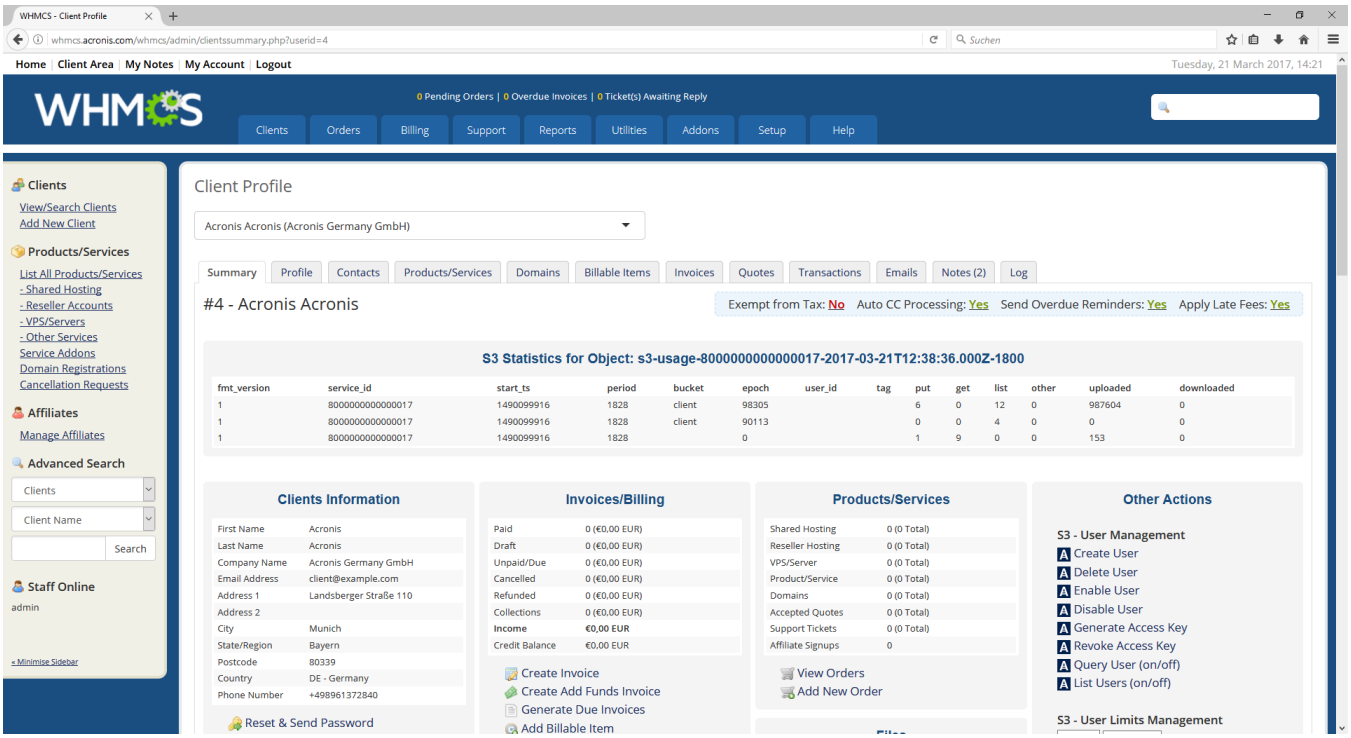


4.4.2 Querying Statistics Objects

You can display usage statistics with the `ostor-usage` service and parameter `obj` specifying the statistics object. WHMCS displays the accessed buckets, user ID, and counters when you click the **Get** button. Create a file `S3_getStatsForObject.php` with the following contents:

```
<?php

// Load configuration and libraries.
require('../../includes/staas_scripts/S3_getConfig.php');
require('../../includes/staas_scripts/S3_requestCurl.php');
require('../../init.php');

// Get s3 statistics object.
function S3_getStatsObjects($object) {

    // Load configuration.
    $s3_config = s3_getConfig();

    // Get s3 statistics object.
    $s3_client = S3_requestCurl(
        $s3_config['s3_key'],
        $s3_config['s3_secret'],
        $s3_config['s3_gateway'],
```

```
        "/?ostor-usage&obj=" . $object,
        "GET"
    );

    // Store s3 result.
    $_SESSION['s3_object_statistic'] = $s3_client;
    $_SESSION['s3_object'] = $object;

    // Redirect back.
    header('Location: ' . $_SERVER['HTTP_REFERER']);
}

// Call function.
S3_getStatsObjects($_GET['object']);

?>
```



4.4.3 Deleting Statistics Objects

You can delete existing statistics objects with the `ostor-usage` service and parameter `obj` specifying the statistics object. WHMCS removes the statistics object from S3 cluster when you click the **Delete** button. Create a file `S3_deleteStatsForObject.php` with the following contents:

```
<?php

// Load configuration and libraries.
require('../../includes/staas_scripts/S3_getConfig.php');
require('../../includes/staas_scripts/S3_requestCurl.php');
require('../../init.php');

// Delete s3 statistics object.
function S3_deleteStatsForObject($object) {

    // Load configuration.
    $s3_config = s3_getConfig();

    // Delete s3 statistics object.
    S3_requestCurl(
        $s3_config['s3_key'],
        $s3_config['s3_secret'],
        $s3_config['s3_gateway'],
        "/?ostor-usage&obj=" . $object,
        "DELETE"
    );

    // Clear array.
    $_SESSION['s3_limits_bucket'] = null;

    // Redirect back.
    header('Location: ' . $_SERVER['HTTP_REFERER']);
}

// Call function.
S3_deleteStatsForObject($_GET['object']);

?>
```