CGRateS
Architectural Components

Dan Christian Bogos
dan.bogos@itsyscom.com

July, 2022

carrier grade realtime charging

Our Background

ITSYSCOM
Engineered Solutions

Located in Bavaria/Germany, over 15 years of experience with architecting server side solutions in VoIP environment

Platform implementations covering both wholesale and retail business categories

Responsibly understanding real-time processing constraints and the seriousness of live system outages

CGRateS
carrier grade realtime charging

About CGRateS

Real-time Enterprise Billing Suite

Pluggable into existing infrastructure
Accommodate new components into ISP/ITSP network (eg: new Comm switch, SMS Service)
Non-intrusive into existing setups

Open Source software

Full sources available on Github repository
No add-ons in private repositories
High consideration for community contributions

Performance Oriented

Built-in advanced cache system (transactional, LRU + TTL records)
Asynchronous processing with micro-threads

Test driven development

Over 5000 tests as part of the test suite

CGRateS
carrier grade realtime charging

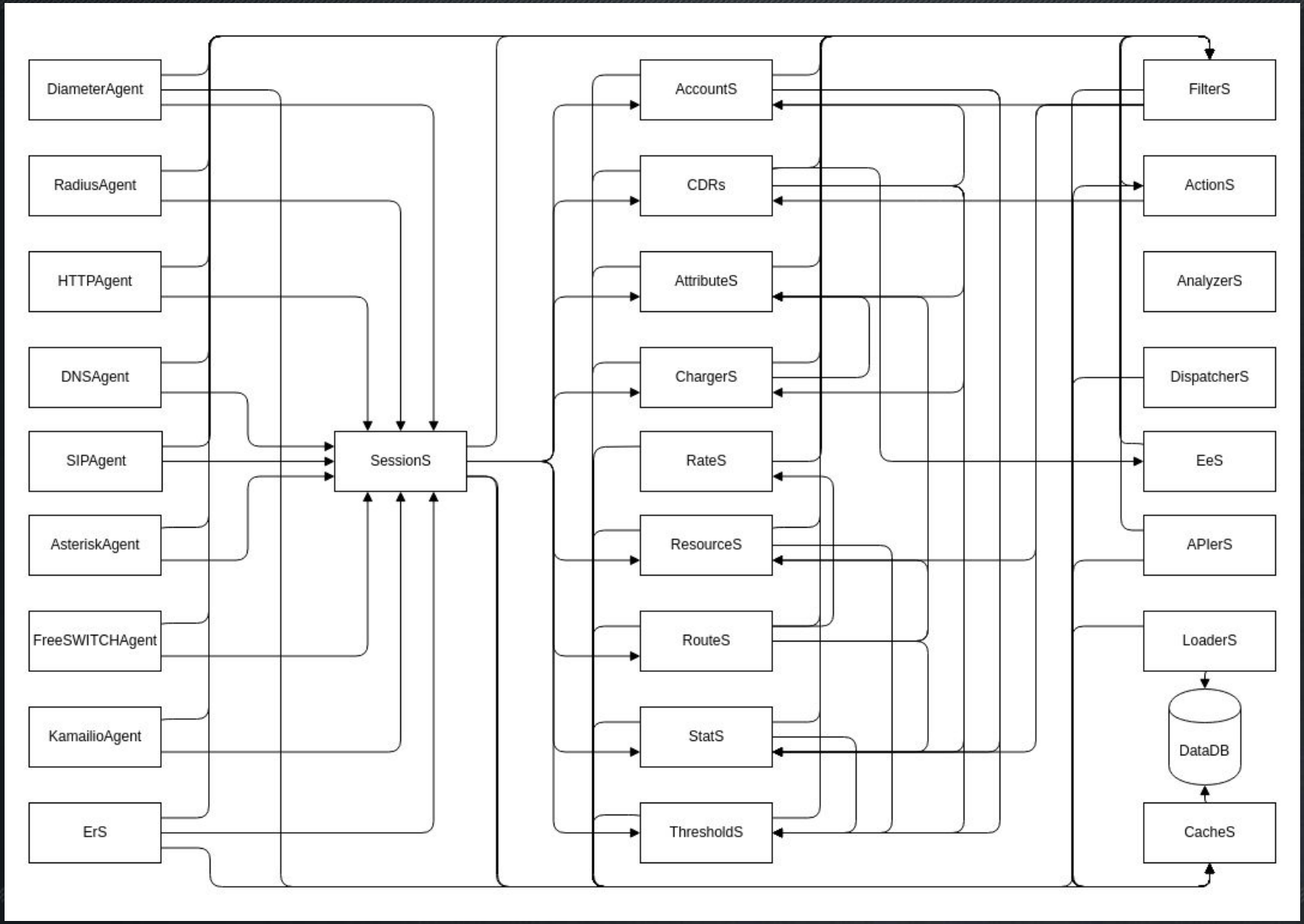
About CGRateS (2)

Modular architecture

Cloud-ready, micro-services with rich set of RPC APIs
Easy to enhance by rewriting specific components

Feature-rich

Online/Offline Charging System (OCS)
Multi-tenancy from day one
Real-time configuration reloads
Rating Engine with Derived Charging and A-Number rating
Account Balances Management with Bundles
Session or Event Charging with balance reservation and refunds
Multi-layer authorization (MaxUsage, AttributesPassword, Routing, ResourceAllocation, STIR/SHAKENIdentity)
CDR logging with support for Interim Records and RatingQueues
Fraud detection with automatic mitigation
Routing/LCR with QoS/Bundles
Event Statistics with pattern monitoring
Diameter/Radius/DNS/SIP Server with process templates (standard agnostic)
Resource allocation controller
API server with GOB, JSON, HTTP-JSON support
Integration with external services (SureTax, APIBan, etc)
Built-in High-Availability and Dynamic-Partitioning support
Agile in developing new features



CGRateS Subsystems interoperability

CGRATES
carrier grade realtime charging

CGRateS binaries

cgr-engine

Services implementation
Controlled via .json or console params

cgr-console

Interface towards the engine
Communicates over JSON-RPC with engine

carrier grade realtime charging

CGRateS binaries (2)

cgr-migrator

Migrates data between versions
Moves data between database types
Controlled via .json or console params

cgr-tester

Load profiling via traffic simulation
Customizable via console parameters

carrier grade realtime charging

CGRateS Data Sources

Cache
LRU/TTL
Chained dependencies between objects
Thread safe
Data replication

DataDB
Performance oriented
Holds active tariff plan data, accounts, stats, management
Supported: Internal, Redis, MongoDB
Remote queries and Replication functionality

CGRateS
carrier grade realtime charging

CacheS

Part of every subsystem

Remote APIs available
Multiple cache partitions
Pre-caching per partition
LRU and TTL support
Groups for dependency removal

APIs for Items

Cache stats
Item IDs
Item expiry time

CRUD operations

Load
Re-load
Flush

CGRateS
carrier grade realtime charging

CGRates Appliances

Online/Offline Charging System

Complex rating
Unlimited bundles
CDR server
High number of interfaces IN / OUT

Routing Server

Flexible routing strategies
Number portability support
Complex rating
Bundles support
Stats/QoS support
Channel limits

CGRATES
carrier grade realtime charging

CGRates Appliances (2)

StatS Server

In memory stat queues for instant access to stats
Predefined statistics used in Telecom (ie: *asr, *acd, *pdd, *ddc, *acc, *tcc)
Generic statistics built on any event fields

Resource utilization control server

Virtual resource counters
Groups of resources

Mediation server

Control Event parameters
Internal or external database

Billing Assurance solution

Passive CDR feed

carrier grade realtime charging

CGRateS Installation

Packaged for Debian

.deb package provided by CGRateS dev team
APT repository hosted by ITsysCOM
Updated based on internal milestones/bug fixes

Packaged for Redhat

.rpm package provided by CGRateS dev team
YUM Repository hosted by ITsysCOM
Updated based on internal milestones/bug fixes

Packaged for Docker

Docker file maintained CGRateS dev team using scratch as image
Docker registry hosted by ITsysCOM
Updated based on internal milestones/bug fixes

Out of sources

Using Go Modules to maintain dependencies
Recommended latest Go stable for compiling

CGRateS Versions

v0.10
Conservative branch, only bug fixes
Production ready
Maintained indefinitely

master
Actively developed
Production ready

1.0
Core modules rewritten
In development
Planned to become the flagship product

carrier grade realtime charging

CGRateS Configuration

Folder .json
Unlimited folders
Unlimited files
Alphabetically ordered
Environment variables

HTTP sources
Multiple HTTP paths supported for one load
Environment variables

DataDB sources
Overwrites the in-memory loaded configuration
Can work in parallel with other sources, always loaded in the end
Overwritten by API calls

carrier grade realtime charging

Configuration multi-files load

CGRATES
carrier grade realtime charging

ConfigS

Serving files from path
Ability to serve files from subfolders

HTTP transport
Both secure and insecure supported

Container optimized
Not parsing environment variables on server side but client one

CGRateS
carrier grade realtime charging

Tariff plans

Online TP

Stored within DataDB

Cached/pre-cached for performance optimization

carrier grade realtime charging

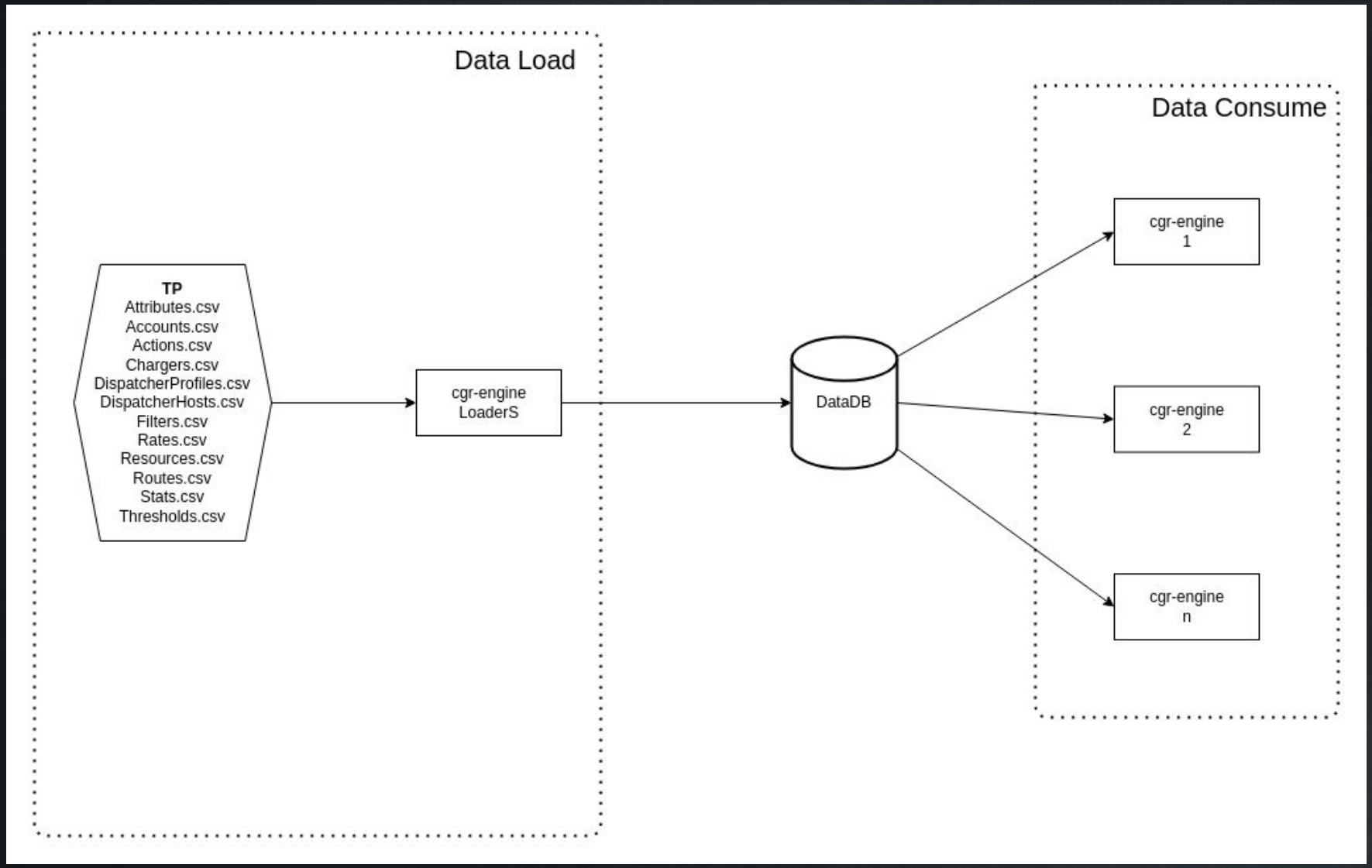
TP loading

APIs
Load from .csv zipped folder
Export from DataDB to zipped folder

LoaderS
Dynamic content via processor templates
Automatic load via linux inotify

Recaching
Localhost by default
Mass-recaching available via DispatcherS

carrier grade realtime charging



CGRateS Data Management

carrier grade realtime charging

Agents

Implementing Comm Switch dialect

Individual agent for each protocol type
Supported dialects: Asterisk, FreeSWITCH, Kamailio, Diameter, Radius, HTTP, DNS, SIP

Protocol agnostic templates

Defined in JSON
Controlling both request as well as replies
Controlling subsystems used via processor flags
RSR field templates and various handlers
Available for Diameter, Radius, HTTP, DNS, SIP

carrier grade realtime charging

RSR Parser

CGRateS private format

Definition: `~*path.FieldName:s/$match_capture_rule/$replace_rule/...{*data_converters}`
Sample: `~*req.Header4:s/(a)/${1}b/:s/(ab)/${1}c/{*duration_seconds&*round:2}`
Dynamic content: `~*req.<~*req.CGRID;~*req.RunID;-Cos>`
Escaped content: `` `>;q=0.7;expires=3600` ``

Data converters

Convert between different types (ie: string to duration)
Multiple converters, own parameters

CGRateS
carrier grade realtime charging

DiameterAgent

Diameter Server implementation

Standard agnostic via processor templates
Transport TCP/SCTP
Synchronous or asynchronous message processing
Per host dictionaries
Default handling of CER/CEA and DWR/DEA messages
Concurrent requests limiting
RAR(ReAuthorizationRequest)/RAA
DPR(DisconnectPeerRequest)/DPA
ASR(AbortSessionRequest

CGRateS
carrier grade realtime charging

RadiusAgent

Radius Server implementation

Standard agnostic via processor templates
Built on https://github.com/cgrates/radigo maintained by CGRateS devs
Transport UDP/TCP
Per host dictionaries

carrier grade realtime charging

HTTPAgent

HTTP Server implementation

Standard agnostic via processor templates
Register handlers on standard HTTP server used by CGRateS
Request encoders: *url, *xml
Reply encoders: *text_plain, *xml

DNSAgent

DNS Server implementation

Standard agnostic via processor templates
Transports: UDP, TCP, TCP-TLS
Supported query types: A, NAPTR

carrier grade realtime charging

SIPAgent

SIP Server implementation

Standard agnostic via processor templates
Transports: UDP, TCP, TCP-TLS
Replies with 302 Redirect always
Support for retransmissions

CGRateS
carrier grade realtime charging

AsteriskAgent

Asterisk ARI client

Built on https://github.com/cgrates/aringo maintained by CGRateS devs
Opening HTTP client + websocket for inbound data
Bidirectional communication opened to SessionS for forced disconnects
Session synchronization with Asterisk
Exchanging channel variables with Asterisk server
Obtaining authorization block via Stasis application + continue once processed
Optionally creating CDR out of CHANNEL_DISTROYED message

FreeSWITCHAgent

FreeSWITCH ESL client

Using https://github.com/cgrates/fsock maintained by CGRateS devs
Bidirectional communication opened to SessionS for forced disconnects
Session synchronization with FreeSWITCH
Exchanging channel variables with FreeSWITCH server
LowBalance/Disconnect warnings
Obtaining authorization block via Park application + dialplan transfer
Optionally creating CDR out of CHAN_HANGUP_COMPLETE message
Extra fields support in CDR

KamailioAgent

Kamailio evapi client

Using https://github.com/cgrates/kamevapi maintained by CGRateS devs
Bidirectional communication opened to SessionS for forced disconnects
Session synchronization with Kamailio
Exchanging pseudovariables with Kamailio server
Predefined events for request/replies

CGRateS
carrier grade realtime charging

SessionS

Accounting Satellite system

Balance reservation in chunks of debit interval
Balance refunds
Debit sleep when needed

Mechanisms handled

Session multi-layer Authorization
Accounting Start/Update/Stop or automated via Start/Stop
Session TTL and synchronization
CDR Generation on-demand
Force disconnect towards CommSwitch
Session forking for DerivedCharging

CGRateS
carrier grade realtime charging

CDR SERVER

Real-time CDR Server
Accessible Internally, GOB, JSON, HTTP-JSON, HTTP-REST interfaces

Advanced functionality
Re-rating
Online / offline CDR exports via EEs interfaces
*dynaprepaid support

Zero configuration CDR sources
Asterisk, FreeSWITCH

CGRateS
carrier grade realtime charging

FilterS

Generic & Adaptive
Processing generic events (hashmaps)

Expandable logic
Each filter type supports own computing logic
`*string, *prefix, *suffix, *cronexp, *rsr, *empty, *exists, *lt, *lte, *gt, *gte, *eq, *ipnet, *apiban, *regex, *never`
Each filter has also the negation variant: *notstring

Performance oriented
Indexed `*string`, `*prefix` types

Dynamic data providers
Ability to query external services
`*accounts, *resources, *stats, *libphonenumber, *asm`

CGRateS
carrier grade realtime charging

FilterS - config options

```
type Filter struct {
     Tenant              string
     ID                  string
     Rules               []*FilterRule
}


type FilterRule struct {
     Type        string
     Element     string
     Values      []string
}
```

Standalone RPC service

Can be used outside of CGRateS scope
Accepting generic events
Using filters as ACL
Full set of APIs available
Isolated subsystem reflecting the platform's performance

Complex rating algorithms

FixedFees, RecurrentFees
MinCost, MaxCost
Configurable RatingUnits and RatingIncrements up nanoseconds
Support for both time units (ie. CallDuration) as well as integer ones (ie: Data, SMSes)
Dynamic weights
Multi-layer rates filtering

RateS (2)

Costs calculator

Based on statically defined Tariff Plans
Cost API call for simulation

Atomic objects

Overwritten by ID on load
Partial updates possible

CGRATES
carrier grade realtime charging

RateS - config options

```
type RateProfile struct {
      Tenant          string
      ID              string
      FilterIDs       []string
      Weights         DynamicWeights
      MinCost         *Decimal
      MaxCost         *Decimal
      MaxCostStrategy string
      Rates           map[string]*Rate
}

type Rate struct {
      ID              string
      FilterIDs       []string
      ActivationTimes string
      Weights         DynamicWeights
      Blocker         bool
      IntervalRates   []*IntervalRate

}

type IntervalRate struct {
      IntervalStart *Decimal
      FixedFee      *Decimal
      RecurrentFee  *Decimal
      Unit          *Decimal
      Increment     *Decimal
}
```

AccountS

Standalone RPC service

Can be used outside of CGRateS scope
Accepting generic events
Using filters as ACL
Full set of APIs available

Flexible Balance configuration

*abstract and *concretes types
Multi-layer filtering
Dynamic Weights support
Dynamic Blockers support

CGRateS
carrier grade realtime charging

AccountS (2)

Management

Direct API calls
AccountProfiles via LoaderS
Scheduler component via ActionS

ThresholdS monitors

Each account update will be reflected on ThresholdS side
ThresholdS can be selected directly from within Account

CGRateS
carrier grade realtime charging

AccountS - config options

```
type Account struct {
      Tenant        string
      ID            string
      FilterIDs     []string
      Weights       DynamicWeights
      Blockers      DynamicBlockers
      Opts          map[string]interface{}
      Balances      map[string]*Balance
      ThresholdIDs  []string
}


type Balance struct {
      ID              string
      FilterIDs       []string
      Weights         DynamicWeights
      Blockers        DynamicBlockers
      Type            string
      Units           *Decimal
      UnitFactors     []*UnitFactor
      Opts            map[string]interface{}
      CostIncrements  []*CostIncrement
      AttributeIDs    []string
      RateProfileIDs  []string
}
```

AccountS - config options (2)

```
type UnitFactor struct {
     FilterIDs []string
     Factor    *Decimal
}
```

```
type CostIncrement struct {
     FilterIDs    []string
     Increment    *Decimal
     FixedFee     *Decimal
     RecurrentFee *Decimal
}
```

ActionS

Standalone Subsystem

Instructions read/ordered/scheduled based on API call
Optimized CPU based on sleep between tasks

Dynamic start/stop via ServiceManager

Can be controlled by NMS

CGRateS
carrier grade realtime charging

ActionS - config options

```
type ActionProfile struct {
      Tenant    string
      ID        string
      FilterIDs []string
      Weights   utils.DynamicWeights
      Blockers  utils.DynamicBlockers
      Schedule  string
      Targets   map[string]utils.StringSet
      Actions []*APAction
}


type APAction struct {
      ID        string
      FilterIDs []string
      TTL       time.Duration
      Type      string
      Opts      map[string]interface{}
      Diktats   []*APDiktat
}


type APDiktat struct {
      Path  string
      Value string
}
```

AttributeS

Standalone RPC service
Can be used outside of CGRateS scope
Accepting generic events
Using layered filters as ACL
Full set of APIs available

Performance optimizations
Indexed filters
AttributeIDs inside API calls for direct selection

Attribute-value store
Unlimited in size
LDAP/Diameter concept

CGRateS
carrier grade realtime charging

AttributeS - config options

```
type AttributeProfile struct {
     Tenant     string
     ID         string
     FilterIDs  []string
     Weights    utils.DynamicWeights
     Blockers   utils.DynamicBlockers
     Attributes []*Attribute
}
```

```
type Attribute struct {
     FilterIDs []string
     Blockers  utils.DynamicBlockers
     Path      string
     Type      string
     Value     config.RSRParsers
}
```

ChargerS

Standalone RPC service

Can be used outside of CGRateS scope
Accepting generic events
Using filters as ACL
Full set of APIs available

Unlimited session/CDR forking

Use cases: reseller/distributors, inbound/outbound, buy/sell prices

Multiple fields replacement

Using AttributeS

CGRateS
carrier grade realtime charging

ChargerS - config options

```
type ChargerProfile struct {
     Tenant       string
     ID           string
     FilterIDs    []string
     Weights      utils.DynamicWeights
     Blockers     utils.DynamicBlockers
     RunID        string
     AttributeIDs []string

}
```

ResourceS

Resource allocation controller
Can be used outside of CGRateS scope
Accepting generic events
Match for multiple profiles, highest prio ID returned
Using filters as ACL
Full set of APIs available

Performance driven architecture
Pre-cached or cached on demand
Asynchronous data backup in offline storage

Rules lifespan
Auto-expiry through UsageUnitTTL

Integrated usage thresholds
Particular case of fraud detection

carrier grade realtime charging

ResourceS - config options

```
type ResourceProfile struct {
     Tenant            string
     ID                string
     FilterIDs         []string
     UsageTTL          time.Duration
     Limit             float64
     AllocationMessage string
     Blocker           bool
     Stored            bool
     Weights           utils.DynamicWeights
     ThresholdIDs      []string

}
```

```
type ResourceUsage struct {
     Tenant     string
     ID         string
     ExpiryTime time.Time
     Units      float64
}
```

StatS

Compute stat metrics for generic events

Internally or remotely accessible
Performance oriented
Using filters as ACL

Multiple Stats Queues

Individual stat queues for same Event processed
Asynchronous data backup in offline storage

Highly configurable StatQueues

Activation interval
QueueLength, TTL, Metrics, Blocker
Static metrics: `*asr, *acd, *tcd, *acc, *tcc, *ddc, *pdd`
Generic metrics: `*sum, *average, *distinct`

ThresholdS support

Particular case of fraud detection

StatS - config options

```
type StatQueueProfile struct {
     Tenant       string
     ID           string
     FilterIDs    []string
     Weights      utils.DynamicWeights
     Blockers     utils.DynamicBlockers
     QueueLength  int
     TTL          time.Duration
     MinItems     int
     Stored       bool
     ThresholdIDs []string
     Metrics      []*MetricWithFilters
}
```

```
type MetricWithFilters struct {
     MetricID  string
     FilterIDs []string
     Blockers  utils.DynamicBlockers
}
```

ThresholdS

Monitor and react on event values

Internally or remotely accessible
Performance oriented
Using filters as ACL

Advanced functionality

Multiple values monitored within a threshold instance
False negatives protection (MinHits)
Action flooding protection (MaxHits, MinSleep)
Asynchronous data backup in offline storage

Actions executed on match

Multiple actions for same threshold matched
Synchronous & Asynchronous Actions executed

CGRateS
carrier grade realtime charging

ThresholdS - config options

```
type ThresholdProfile struct {
     Tenant           string
     ID               string
     FilterIDs        []string
     MaxHits          int
     MinHits          int
     MinSleep         time.Duration
     Blocker          bool
     Weights          utils.DynamicWeights
     ActionProfileIDs []string
     Async            bool
}
```

RouteS

Core component logic
Internally or remotely accessible through APIs
Non-intrusive, information retrieved via RPC
Using filters as ACL

Tightly coupled with ACCOUNTING subsystem
Provides LCR over bundles

Integrates traffic patterns
Computes LCR for specific call duration

Extended functionality through multiple strategies
*weight, *lc, *hc, *qos, *reas, *reds, *load
Flexible strategy parameters

CGRateS
carrier grade realtime charging

RouteS - config options

```
type RouteProfile struct {
     Tenant             string
     ID                 string
     FilterIDs          []string
     Weights            utils.DynamicWeights
     Blockers           utils.DynamicBlockers
     Sorting            string
     SortingParameters  []string
     Routes             []*Route
}


type Route struct {
     ID                string
     FilterIDs         []string
     AccountIDs        []string
     RateProfileIDs    []string
     ResourceIDs       []string
     StatIDs           []string
     Weights           utils.DynamicWeights
     Blockers          utils.DynamicBlockers
     RouteParameters   string
}
```

DispatcherS

Standalone service
Remotely accessible through APIs
Transparent implementation of RPC methods for the supported subsystems

Request router
Generic filters for matching/queries
Using hashmap events with dynamic type fields
Unlimited number of host categories with failover on hosts category

Dynamic Hosts
Fully configurable by API
Connections established on the fly
Cached connections

CGRateS
carrier grade realtime charging

DispatcherS (2)

Flexible routing strategies
*weight, *random, *round_robin, *broadcast

API security
AttributeS as data provider
***apiKey** for authentication
APIMethods for authorization

Routing path cache
Using optional ***routeID** as option
Only first request is dynamically routed

Registrar
Allowing Hosts to register from remote (avoiding the need of knowing their IP details)
Elastic routing

carrier grade realtime charging

DispatcherS - config options

```
type DispatcherProfile struct {
      Tenant              string
      ID                  string
      Subsystems          []string
      FilterIDs           []string
      ActivationInterval *utils.ActivationInterval
      Strategy            string
      StrategyParams      map[string]interface{}
      Weight              float64
      Hosts               DispatcherHostProfiles
}

type DispatcherHost struct {
      Tenant string
      ID                    string
      Address               string
      Transport             string
      ConnectAttempts       int
      Reconnects            int
      MaxReconnectInterval  time.Duration
      ConnectTimeout        time.Duration
      ReplyTimeout          time.Duration
      TLS                   bool
      ClientKey             string
      ClientCertificate     string
      CaCertificate         string
}
```

ErS

Online/offline Event reader service

Using linux inotify for online imports
Scheduled reads in case of offline processing

Multiple interfaces

```
*file_csv, *partial_csv, *file_xml, *flatstore
*kafka_json_map, *amqp_json_map,  *amqpv1_json_map, *s3_json_map, *sqs_json_map
*sql
```


CGRateS
carrier grade realtime charging

EeS

Online/offline Event exporter service

Called by CDRs in case of online exports
Called by APIs in case of offline exports

Multiple interfaces

```
*file_csv, *file_fwv, *virt
*http_post, *http_json_map,
*amqp_json_map, *amqpv1_json_map, *sql_json_map, *kafka_json_map, *sqs_json_map
*elastic
```


carrier grade realtime charging

Fraud Mitigation

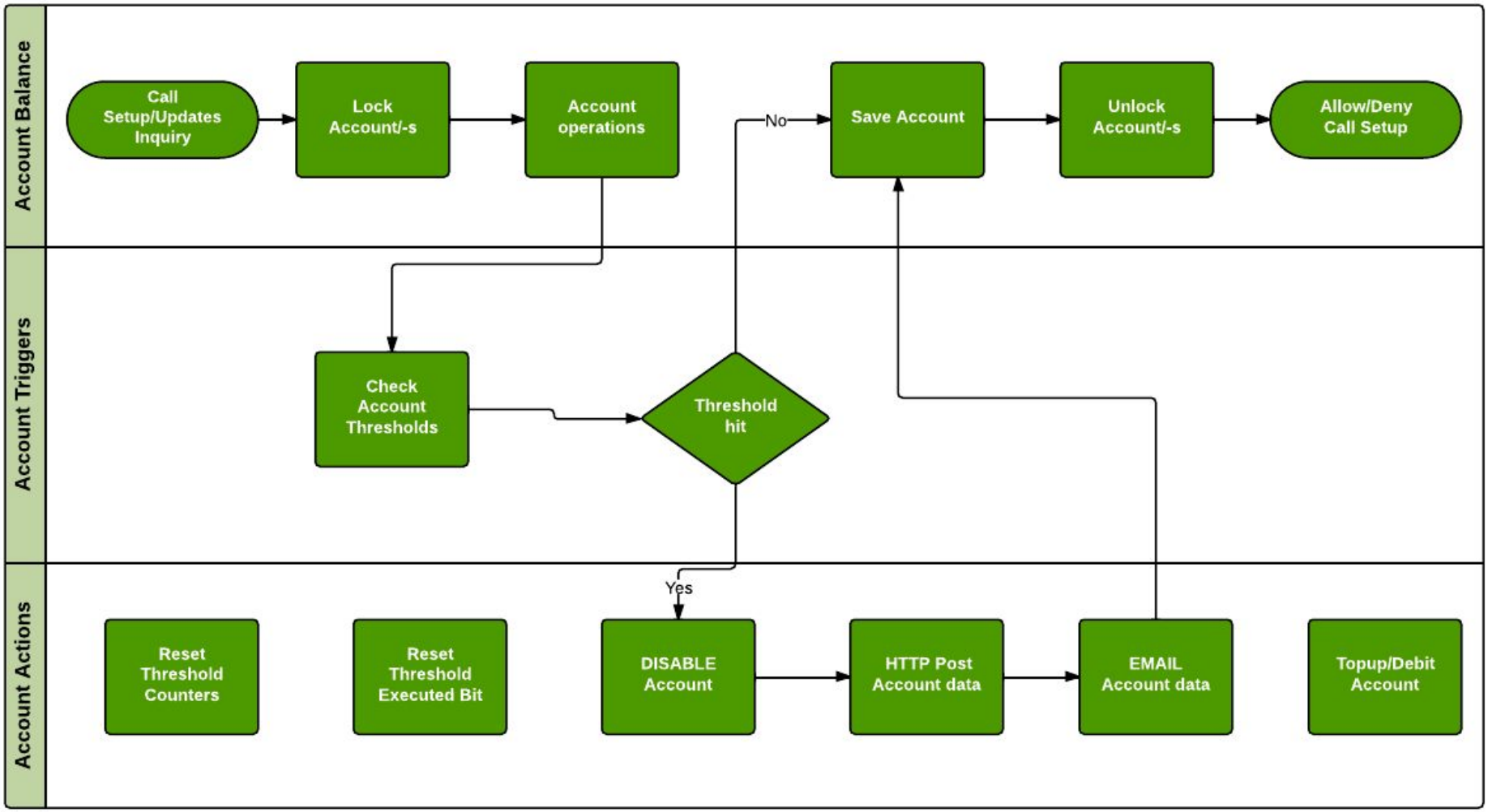
Part of Accounting

Tightly integrated, balance operations cannot avoid it
Min/Max Balance and Counter thresholds

Part of ThresholdS

Monitoring Accounts/Balances
Monitoring CDRs
Monitoring StatS
Monitoring ResourceS

carrier grade realtime charging



Account operations

RATES
carrier grade realtime charging

CGRates Peripherals

APIer (RPC server)

Tariff plan and Account management
Export commands form internal components (Eg: get_cdrs, export_cdrs, etc)
Partial and full rates/accounts reload without restarts

Console

Interactive and non-interactive
History
Help
Command auto-completion

Loader

CSV Imports

Tester

carrier grade realtime charging

APIER SERVER

RPC Server functionality
GOB-RPC
JSON-RPC over socket, websocket, HTTP

Used by console and other GUI-like components to interact with the core

Rich set of remote methods available
Own folder in sources for auto-documenting

carrier grade realtime charging

APIER SERVER (2)

Tariff plan management
Partial and full rates reload without restarts
CSV imports

Realtime costs and account management
Manual add/debit actions
Query costs and accounts status

Operational commands
Used resources
Check StatS

CGRateS
carrier grade realtime charging

Questions?

Website
http://www.cgrates.org

Documentation
http://cgrates.readthedocs.org

Code + issues tracker
https://github.com/cgrates/cgrates

Support
Google group: **CGRateS**
IRC Freenode: **#cgrates**

CGRateS
carrier grade realtime charging