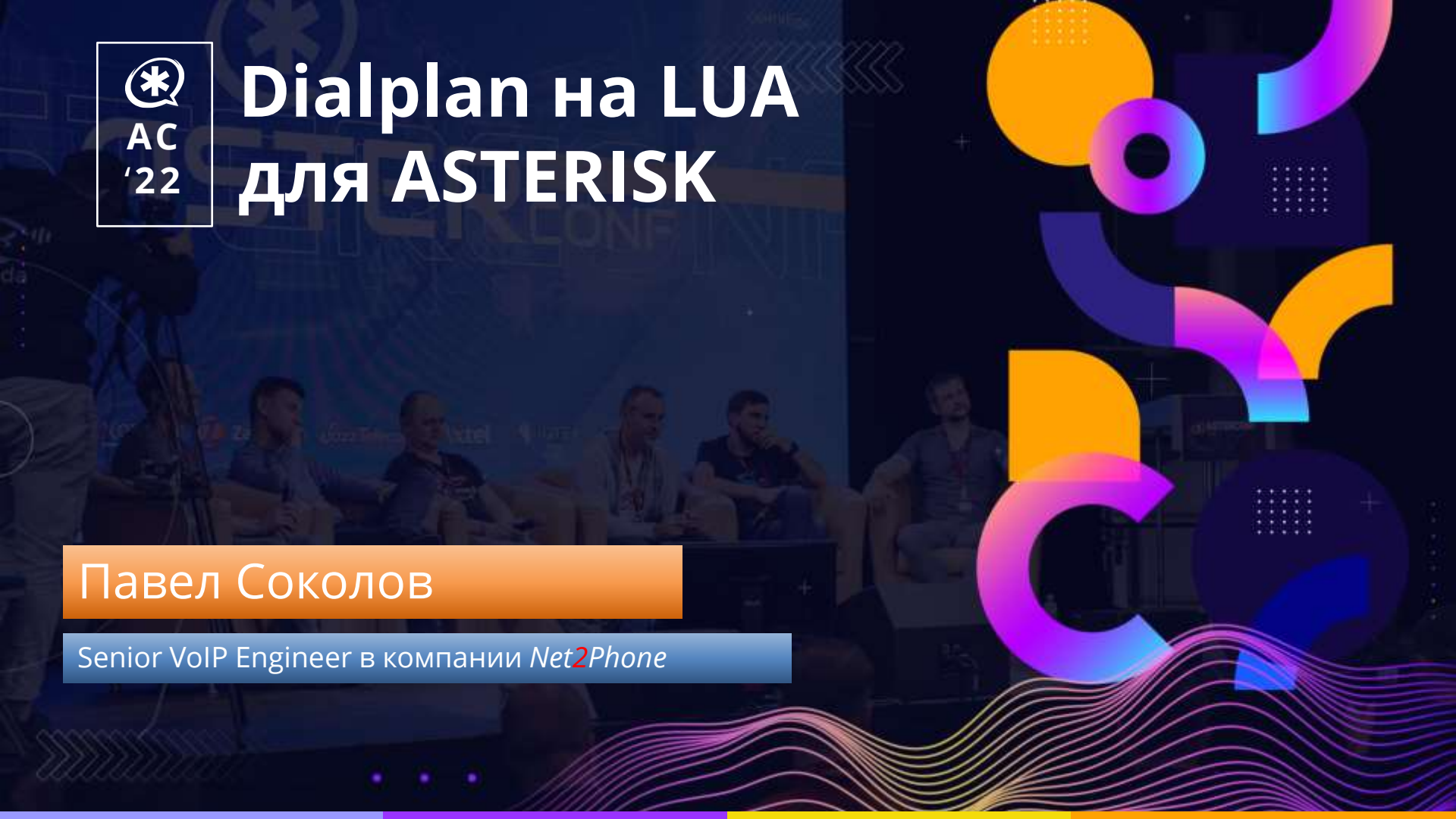
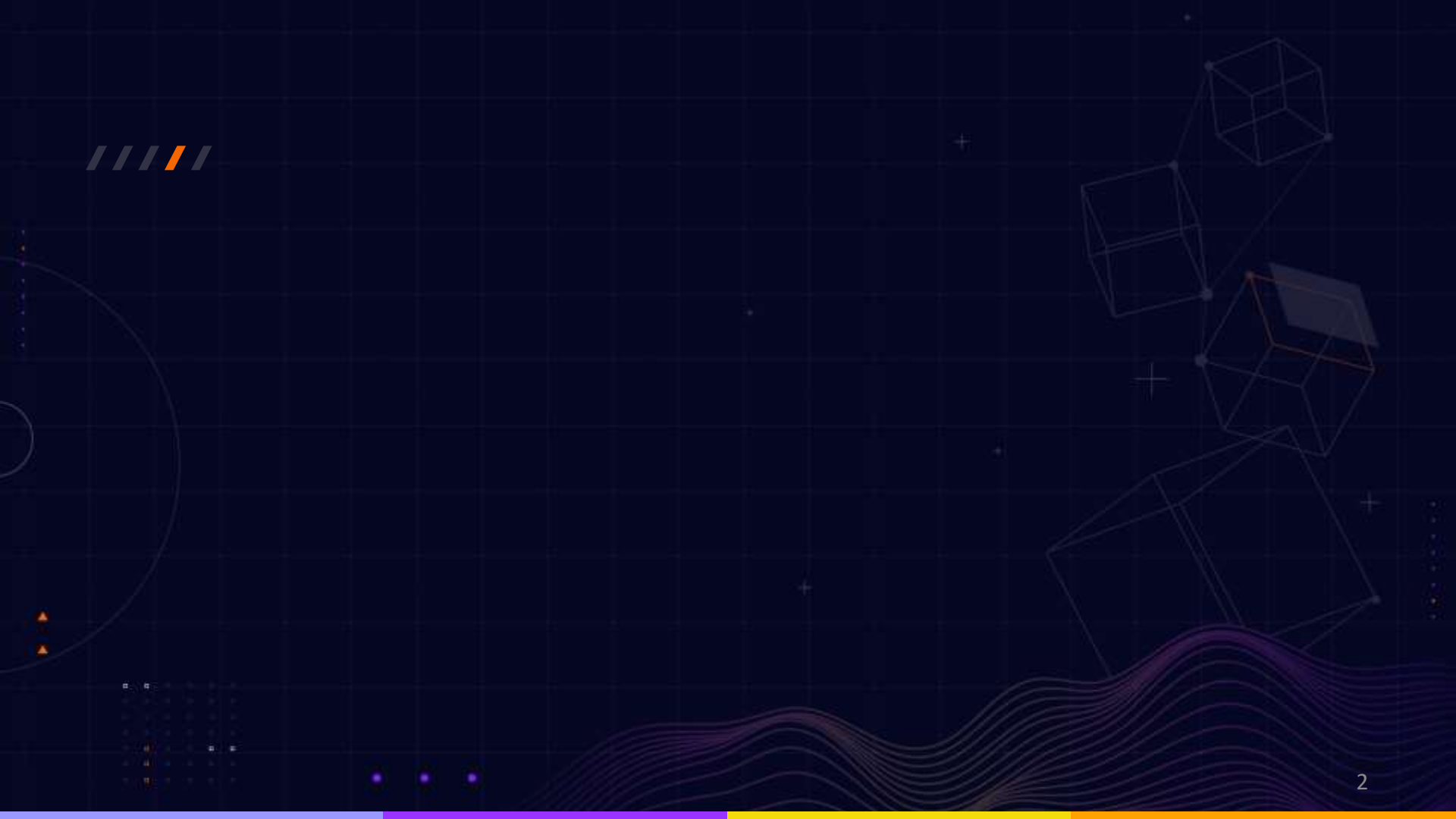


Dialplan на LUA для ASTERISK

Павел Соколов

Senior VoIP Engineer в компании *Net2Phone*



Коротко о себе

- В данный момент работаю в компании Net2Phone в должности Senior VoIP Engineer.
- Начал работать с Asterisk в 2011 году.
- 99% выполненных проектов на базе «ванильного» или «чистого» Asterisk.
- Пишу dialplan на Lua для Asterisk с 2016 года
- Так же использую Lua в Openresty (nginx)

Павел Соколов

AC '22

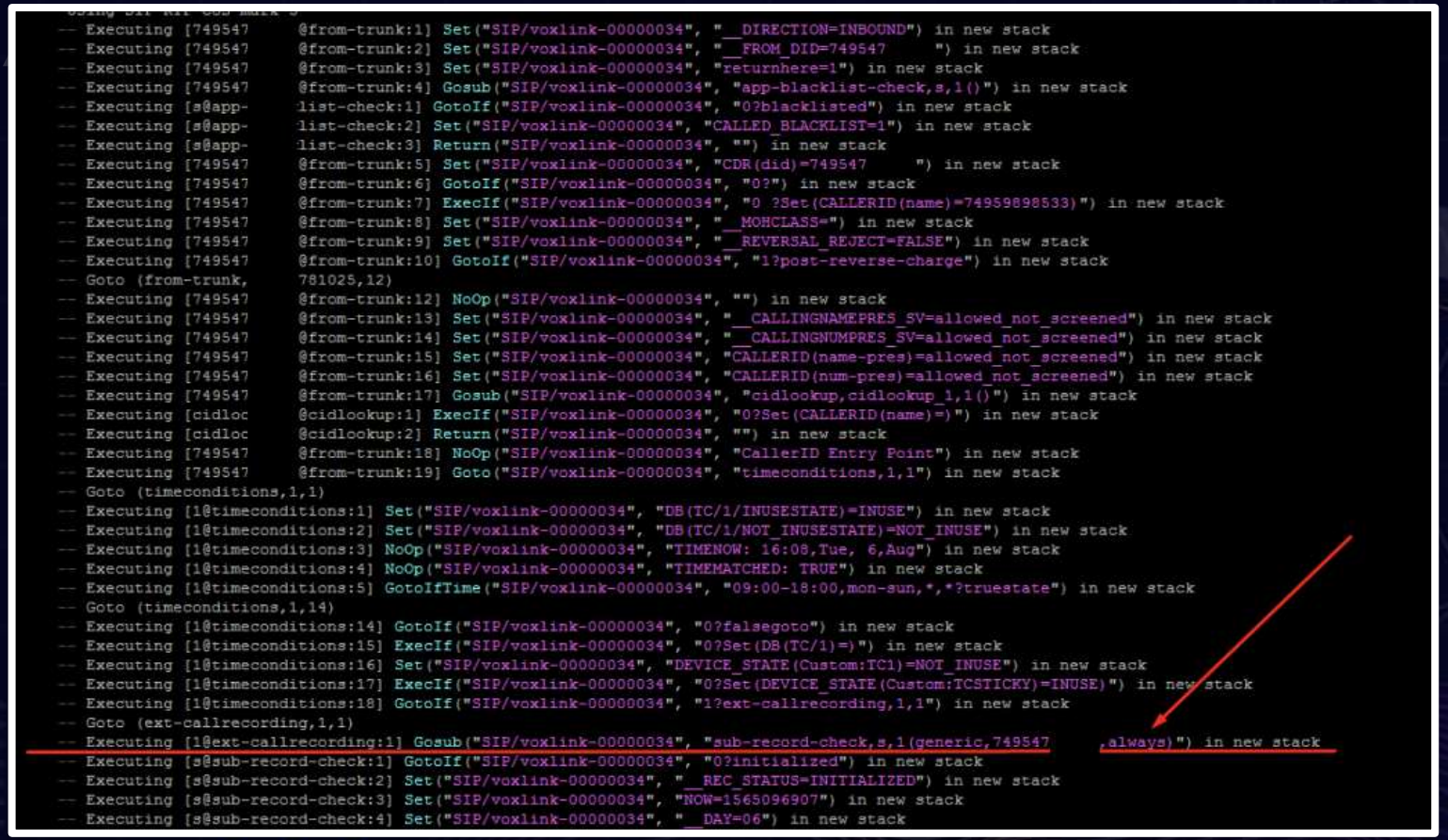
Предисловие

Предисловие

```
same => n,GotoIf($[ "${DIALSTATUS}" = "ANSWER" ]?call_answer)
same => n,Set(__mcount=$[ ${mcount} + 1])
same => n,Goto(rotation_loop)
same => n,Goto(continue)
;
;
same => n(continue),NoOP(-->continue)
same => n,Set(__message=shoulder=B&action=stopstep&id=${id}&priority=${priority_${step}}&step=${step})
same => n,GoSub(sendTrackPostInfo,s,1)
same => n,ContinueWhile()
;
;
same => n(exit_while),NoOP(--> exit while)
same => n,Set(__message=shoulder=B&action=stopstep&id=${id}&priority=${priority_${step}}&step=${step})
same => n,GoSub(sendTrackPostInfo,s,1)
same => n,ExitWhile()
same => n,NoOP(--> end while)
same => n,Set(__message=shoulder=B&action=stopstep&id=${id}&priority=${priority_${step}}&step=${step})
same => n,GoSub(sendTrackPostInfo,s,1)
same => n,EndWhile
same => n(p_busy),NoOP( play busy manager_busy_rec:${manager_busy_rec} manager_busy_wav:${manager_busy_wav})
same => n,NoOp(${queue})
same => n,Set(QueueBusy = 1)
same => n,ExecIf($[ ${LEN(${queue})} > 1 ]?Goto(queue-context,s,1):NoOP(nothing))
same => n,ExecIf($[ ${LEN(${manager_busy_wav})} > 1 ]?Progress():NoOP(nothing))
same => n,ExecIf($[ ${LEN(${manager_busy_wav})} > 1 ]?Playback(/opt/background/${manager_busy_wav}):NoOP(nothing)
same => n,StopMixMonitor()
same => n,ExecIf($[ "${manager_busy_rec}" = "1" ]?Set(ARRAY(__r_year,__r_month,__r_day)=${STRFTIME(${EPOCH},,%Y)}
same => n,ExecIf($[ "${manager_busy_rec}" = "1" ]?Answer():NoOP(nothing))
same => n,ExecIf($[ "${manager_busy_rec}" = "1" ]?Record(/var/spool/asterisk/monitor/${id}.wav,15,300,xk):NoOP(no
same => n,Hangup()
same => n(call_answer),Hangup()
```

AC '22

Предисловие



AC '22

Недостатки extensions.conf

Недостатки extensions.conf

- Стандартный dialplan сложен с точки зрения восприятия/чтения кода;

Недостатки extensions.conf

- Стандартный dialplan сложен с точки зрения восприятия/чтения кода;
- в стандартном dialplan нет настоящих функций;

AC '22

Недостатки extensions.conf

- Стандартный dialplan сложен с точки зрения восприятия/чтения кода;
- в стандартном dialplan нет настоящих функций;
- в стандартном dialplan нет типов данных;

Недостатки extensions.conf

- Стандартный dialplan сложен с точки зрения восприятия/чтения кода;
- в стандартном dialplan нет настоящих функций;
- в стандартном dialplan нет типов данных;
- в стандартном dialplan нет структур данных;

AC '22

Вариант решения от разработчиков Asterisk

Вариант решения от разработчиков Asterisk

- Где-то в далеком 2005 году они предложили нам AEL

```
macro ael-std-exten-ael( ext , dev ) {
        Dial(${dev}/${ext},20);
        switch(${DIALSTATUS}) {
        case BUSY:
                Voicemail(${ext},b);
                break;
        default:
                Voicemail(${ext},u);
        };
        catch a {
                VoiceMailMain(${ext});
                return;
        };
    return;
};
```

```
    Background(demo-moreinfo);
    goto s|instructions;
};
3 => {
    Set(LANGUAGE()=fr);
    goto s|restart;
};
1000 => {
    goto ael-default|s|1;
};
500 => {
    Playback(demo-abouttotry);
    Dial(IAX2/guest@misery.digium.com/s@default);
    Playback(demo-nogo);
    goto s|instructions;
};
600 => {
    Playback(demo-echotest);
    Echo();
    Playback(demo-echodone);
    goto s|instructions;
};
```

AC '22

Чем нам помог AEL?

Чем нам помог AEL

- ~~Стандартный dialplan сложен с точки зрения восприятия/чтения кода;~~
- в стандартном dialplan нет реальных функций;
- в стандартном dialplan нет типов данных;
- в стандартном dialplan нет структур данных;

AC '22

В 2008 году на Astricon Мэттью Николсон представил модуль pbx_lua

AC '22

Что такое LUA

- Lua (лу́а, с порт. — «луна») — скриптовый язык программирования, разработанный в подразделении Tecgraf (Computer Graphics Technology Group) Католического университета Рио-де-Жанейро (Бразилия).
- Авторы языка Роберту Иерузалимски, Валдемар Селиш, Луиш Энрике ди Фигейреду
- По идеологии и реализации язык Lua ближе всего к JavaScript, в частности, он также реализует прототипную модель ООП, но отличается паскалеподобным синтаксисом и более мощными и гибкими конструкциями.

Что такое LUA

- Характерной особенностью Lua является реализация большого числа программных сущностей минимумом синтаксических средств.

- Все составные пользовательские типы данных (массивы, структуры, множества, очереди, списки) реализуются через механизм таблиц, а механизмы объектно-ориентированного программирования, включая множественное наследование — с использованием метатаблиц.

Где используется кроме Asterisk

- Kamailio
- FreeSWITCH
- Nginx (рекомендую OpenResty)
- Adobe Lightroom
- В компьютерных играх (WoW, S.T.A.L.K.E.R., Garry's Mod и др.)

AC '22

Где используется кроме Asterisk

- Пакетный менеджер RPM содержит встроенный интерпретатор Lua
- Nmap, WireShark (lua API)
- Tarantool
- QUIK— российский трейдерский терминал для доступа к биржевым торгам.
- И в многих других программных продуктах.

AC '22

RTFM по языку Lua

- https://www.lua.org
- https://lua.org.ru
- https://tylerneylon.com/a/learn-lua/ - Learn Lua in 15 Minutes

Книга «Программирование на языке Lua»

Автор:Роберту Иерузалимски

Спрашивайте в книжных магазинах вашего города ☺

RTFM по языку Lua для Asterisk

- https://wiki.asterisk.org/wiki/display/AST/Lua+Dialplan+Configuration

AC '22

pbx_lua: модуль для Asterisk

- Модуль представлен в 2008 году на Astricon автором Мэттью Николсоном из Digium, Inc. (Matthew Nicholson mnicholson@digium.com)

AC '22

pbx_lua: цели создания

Из доклада автора :

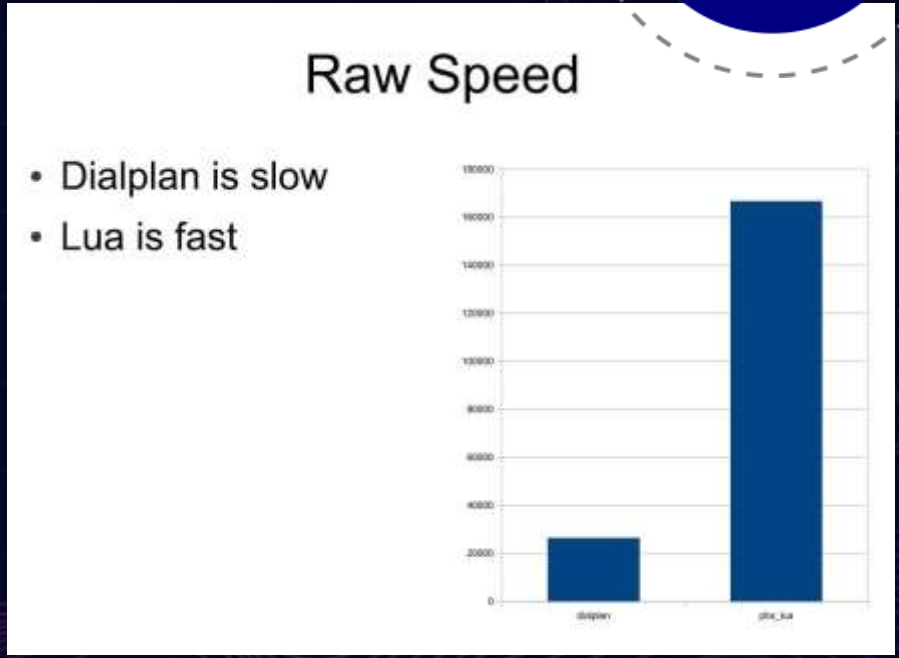
- Удобное конфигурирования PBX систем;
- интеграции с БД;
- красивые(шикарные) скрипты обработки звонков;
- интеграция с WEB-сервисами;
- конечно же для того, для чего вы используете extensions.conf.

AC '22

pbx_lua: цели создания

Из доклада автора :

- Lua прост
- В нем есть функции
- В нем есть типы данных
- В нем есть структуры данных
- Lua расширяем (Luarocks)



pbx_lua: start/restart/reload

pbx_lua: start/restart/reload

- Asterisk загружает файл extensions.lua с диска в память один раз
- Asterisk ожидает, что при загрузке в этом файле будет найдена глобальная таблица с именем «extensions»

pbx_lua: start/restart/reload

- Если в файле есть ошибки, будет сообщено об ошибках и загруженный до этого extensions.lua останется в памяти не перезаписаным
- Каналы, которые существовали до того, как была выдана команда перезагрузки, также будут продолжать использовать загруженный extensions.lua

AC '22

pbx_lua: start/restart/reload

- extensions.lua можно перезагрузить, перезагрузив модуль pbx_lua.

**CLI> module reload pbx_lua.so*

pbx_lua: выполнение extensions.lua при звонке

- extensions.lua считывается **из памяти** и выполняется один раз для каждого канала, который выполняет расширение на основе Lua.
- Дополнительные модули/файлы подключаемые с помощью ***require*** перечитываются с диска после чтения extensions.lua из памяти
- Ошибки во время выполнения регистрируются, и канал, на котором произошла ошибка, отключается.

AC '22

pbx_lua: Пример таблицы extensions

```
1  extensions = {
2      ["contextName"] = {
3          ["100"] = functionN1;
4          ["_10X"] =  function(context, extension)
5                          app.dial("SIP/"..extension, 60)
6                      end;
7          ["h"] = function () app.Hangup(16) end;
8      };
9  }
```

Функция для exten идет с priority = 1. Остальных приоритетов нет!

AC '22

pbx_lua: Переменные

Теперь у нас «два набора переменных»:

- Переменные Lua:

```lua
local i = 0
local cidName = "Operator 1"

local configRedis = {
    server      = "127.0.0.1",
    port        = 6379,
    expire      = 5400,
    delay       = 2
}
```

pbx_lua: Переменные

Теперь у нас «два набора переменных»:

- Переменные канала Asterisk доступны через функцию *channel* :

```
10  -- Пример чтения из канальной в переменную Lua
11  local uniq = channel["UNIQUEID"]:get()
12
13  -- Пример записи в канальную перменную
14  channel["CALLERID(num)"]:set("74951231122")
15  local cidName = "Operator 1"
16  channel["CALLERID(name)"]:set(cidName)
```

pbx_lua: Приложения Asterisk

Доступны с помощью функции(таблицы) ***app***:


```
18  -- Пример вызова приложений Asterisk
19  app.dial("SIP/100", nil, "m")
20  app.Dial(dialString, dialTimeoute, options)
21  app.Hangup()
22  app.Goto(context,exten,1)
```

AC '22

pbx_lua: Пример простого extensions.lua

```lua
--
-- Пример простого dialplan
--
function main( context, extension )
    --
    app.Answer()
    --
    app.Playback("tt-monkeys")
    --
    app.Hangup()
    --
end
--
extensions = {
    ["default"] = {
        ["_[a-Z0-9]."] = main;
        ["h"] = function () app.Hangup(16) end;
    };
}
```

AC '22

extensions conf vs lua: Ветвления

AC '22

extensions conf vs lua: Ветвления

```
same => n,GotoIf($[ "${QueueBusy}" = "1" ]?play:noplay)
same => n,Goto(noplay)
same => n(play),Playback(queue-periodic-announce)
same => n(noplay),Queue(${queue},tTxX,,,,,,queue-answered)
```

extensions conf vs lua: Ветвления

```
same => n,GotoIf($[ "${QueueBusy}" = "1" ]?play:noplay)
same => n,Goto(noplay)
same => n(play),Playback(queue-periodic-announce)
same => n(noplay),Queue(${queue},tTxX,,,,,,queue-answered)
```

```
if QueueBusy=="1" then
    --
    app.Playback("queue-periodic-announce")
    --
end
--
app.Queue(queue,"tTxX,,,,,,queue-answered")
```

extensions conf vs lua: Ветвления

```
same => n,GotoIf($[ "${QueueBusy}" = "1" ]?play:noplay)
same => n,Goto(noplay)
same => n(play),Playback(queue-periodic-announce)
same => n(noplay),Queue(${queue},tTxX,,,,,,queue-answered)
```

```
--
if QueueBusy=="1" then app.Playback("queue-periodic-announce") end
--
app.Queue(queue,"tTxX,,,,,,queue-answered")
--
```

extensions conf vs lua: Циклы

```
same => n,ExecIf($[ ${LEN(${ivr_calltrack})} > 0 ]?Goto(callTrackIVR1,s,1):NoOP(nothing))
same => n,While($["${SET(__step=${SHIFT(steps,-)})}" != ""])
same => n,NoOp(step:${step})
same => n,Set(__message=shoulder=B&action=startstep&id=${id}&priority=${priority_${step}}&step=${step})
same => n,GoSub(sendTrackPostInfo,s,1)
;;;
```

AC '22

extensions conf vs lua: Циклы

```
same => n,ExecIf($[ ${LEN(${ivr_calltrack})} > 0 ]?Goto(callTrackIVR1,s,1):NoOP(nothing))
same => n,While($["${SET(__step=${SHIFT(steps,-)})}" != ""])
same => n,NoOp(step:${step})
same => n,Set(__message=shoulder=B&action=startstep&id=${id}&priority=${priority_${step}}&step=${step})
same => n,GoSub(sendTrackPostInfo,s,1)
;;;
```

```
same => n(continue),NoOP(-->continue)
same => n,Set(__message=shoulder=B&action=stopstep&id=${id}&priority=
same => n,GoSub(sendTrackPostInfo,s,1)
same => n,ContinueWhile()
```

extensions conf vs lua: Циклы

```
same => n,ExecIf($[ ${LEN(${ivr_calltrack})} > 0 ]?Goto(callTrackIVR1,s,1):NoOP(nothing))
same => n,While($["${SET(__step=${SHIFT(steps,-)})}" != ""])
same => n,NoOp(step:${step})
same => n,Set(__message=shoulder=B&action=startstep&id=${id}&priority=${priority_${step}}&step=${step})
same => n,GoSub(sendTrackPostInfo,s,1)
;;;
```

```
same => n(continue),NoOP(-->continue)
same => n,Set(__message=shoulder=B&action=stopstep&id=${id}&priority=
same => n,GoSub(sendTrackPostInfo,s,1)
same => n,ContinueWhile()
```

```
same => n(exit_while),NoOP(--> exit while)
same => n,Set(__message=shoulder=B&action=sto
same => n,GoSub(sendTrackPostInfo,s,1)
same => n,ExitWhile()
```

AC '22

extensions conf vs lua: Циклы

```
while marker do
    --
    result_table[counter] =  channel["ODBC_FETCH("..func_odbc_id..")"]:get()
    --
    if type(result_table[counter])=="table" then
        --
        pr_table(result_table[counter],"router")
        --
    else
        --
        app.Log("NOTICE", ps.."row #"..counter..": <"..tostring(result_table[counter])..">")
        --
    end
    --
    counter = counter+1
    --
    if counter>rows then
        marker = false
    end
    --
end
```

extensions conf vs lua: Циклы


```lua
for i = 1, lenDsTable, 1 do
    --
    app.Log("NOTICE", ps.."index= "..i.." Dial (_"..dsTable[i].dial_string.." , "..dialTimeout.." , "..dialOptions.." _)")
    --
    app.Dial(dsTable[i].dial_string, dialTimeout, dialOptions)
    --
    dial_strings_marker = true
    --
    if channel.DIALSTATUS:get()=="ANSWER" or channel.DIALSTATUS:get()=="CANCEL" then
        --
        break
        --
    end
    --
end
```

```lua
for key,value in pairs(faxopt) do
    --
    channel.FAXOPT(key):set(value)
    --
end
```

AC '22

extensions conf vs lua: Типы данных

В Lua есть 8 типов:

- number(число)
- string(строка)
- boolean(логический тип)
- nil (тип "ничего")
- table,
- function,
- userdata,
- thread.

extensions conf vs lua: Киллер фича

AC '22

extensions conf vs lua: Киллер фича

extensions.lua можно скомпилировать. Это повысит производительность, так как pbx_lua больше не нужно будет анализировать extensions.lua при загрузке.

```
$ mv extensions.lua extensions.lua.orig
$ luac -o extensions.lua.orig extensions.lua
```

AC '22

extensions conf vs lua: Киллер фича 2

Заменить pbx_lua на luajit (Just-In-Time Compiler for Lua) при компиляции Asterisk.

AC '22

extensions conf vs lua: Киллер фича 2

Заменить pbx_lua на luajit (Just-In-Time Compiler for Lua) при компиляции Asterisk.

Цитата от Андрея Ярина @ShadoWalkeR30:

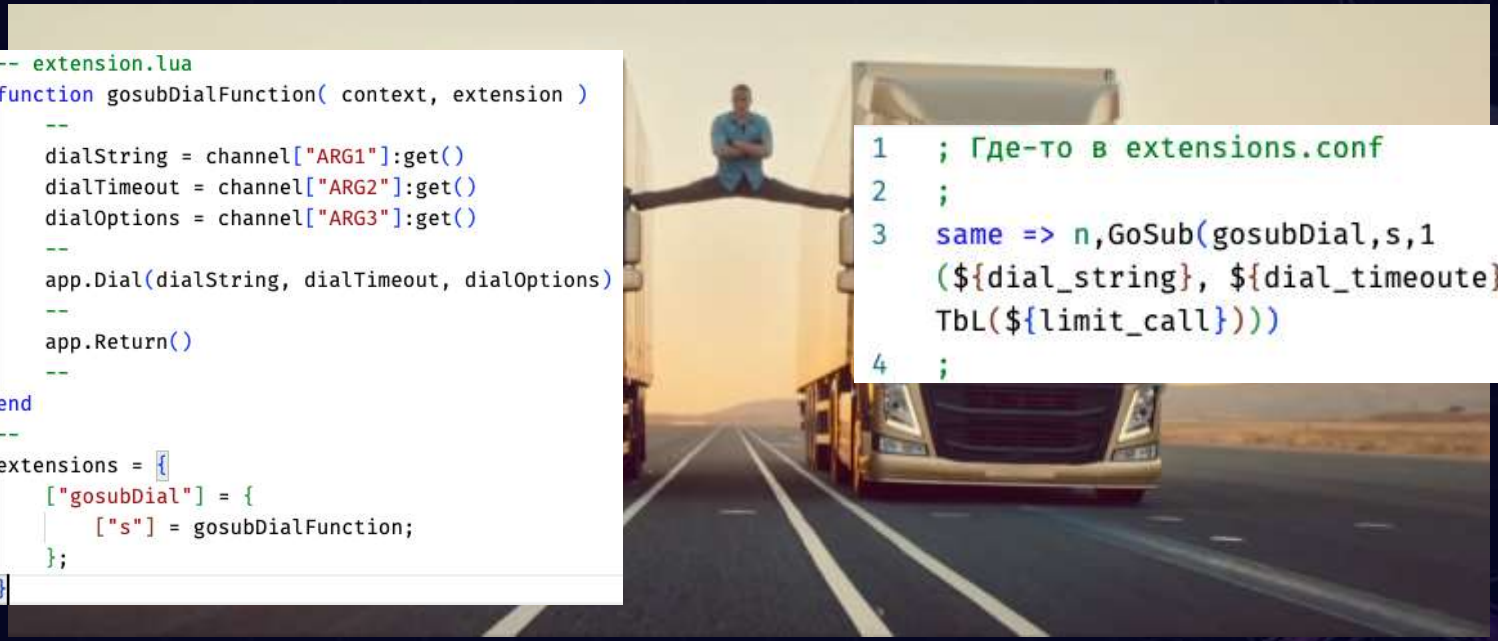
«копируешь luajit модуль в исходники. Ставишь хедеры от luajit, правишь инклуды и в menuconfig добавляешь руками...»

AC '22

pbx_lua: Пример интеграции extensions .lua и .conf

pbx_lua: Пример интеграции extensions .lua и .conf

```lua
-- extension.lua
function gosubDialFunction( context, extension )
    --
    dialString = channel["ARG1"]:get()
    dialTimeout = channel["ARG2"]:get()
    dialOptions = channel["ARG3"]:get()
    --
    app.Dial(dialString, dialTimeout, dialOptions)
    --
    app.Return()
    --
end
--
extensions = {
    ["gosubDial"] = {
        ["s"] = gosubDialFunction;
    };
}
```



```
; Где-то в extensions.conf
;
same => n,GoSub(gosubDial,s,1
(${dial_string}, ${dial_timeoute},
TbL(${limit_call})))
;
```

Luarocks

- Пакетный менеджер библиотек для Lua (https://luarocks.org)
- Пример установки библиотеки из luarocks:

```
# luarocks install lua-cjson
```

- Подключение библиотеки в коде

```
local cjson = require "cjson"
```

AC '22

Luarocks

Например я использовал следующие библиотеки:

Имя модуля	Описание
lua-cjson	C библиотека для работы с JSON
redis-lua	Работа с Redis
net-url	Для парсинга и построения URL
httpclient	для работы с запросами HTTP
luasql-mysql	Интерфейс для работы с БД

pbx_lua: Работа с БД

AC '22

pbx_lua: Работа с БД

КТО-ТО: звонит

AC '22

pbx_lua: Работа с БД

КТО-ТО: звонит

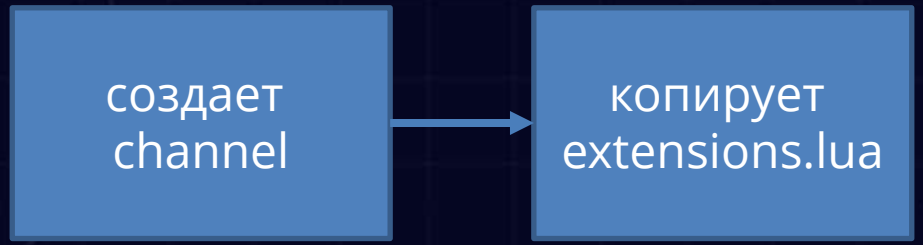
ASTERISK:

создает channel

pbx_lua: Работа с БД

КТО-ТО: звонит

ASTERISK:



pbx_lua: Работа с БД

КТО-ТО: звонит

ASTERISK:

pbx_lua: Работа с БД

КТО-ТО: звонит

ASTERISK:

pbx_lua: Работа с БД

Два других способа работы с БД:

- Запросами HTTP в бэкенд (с кешированием, и прочими куртизанками)

pbx_lua: Работа с БД

Два других способа работы с БД:

- Запросами HTTP в бэкенд (с кешированием, и прочими куртизанками)
- Используй func_odbc!

AC '22

pbx_lua: Работа с БД

Два других способа работы с БД:

- Запросами HTTP в бэкенд (с кешированием, и прочими куртизанками)
- Используй func_odbc!

...но это не точно ☺

pbx_lua: Работа с БД

Преимущества func_odbc:

- Одно постоянное подключение к БД

pbx_lua: Работа с БД

Преимущества func_odbc:

- Одно постоянное подключение к БД
- Контролируется ядром Asterisk

AC '22

pbx_lua: Работа с БД

func_odbc.conf:

```
[SQL]
dsn=mysql1
readsql=${ARG1}

[SQLmulti]
dsn=mysql1
mode=multirow
readsql=${ARG1}
```

AC '22

pbx_lua: Работа с БД

func_odbc.conf:

```
[SQL]
dsn=mysql1
readsql=${ARG1}

[SQLmulti]
dsn=mysql1
mode=multirow
readsql=${ARG1}
```

extensions.lua:

```
1  -- example
2  local exten = "101"
3  local QueryString = "select username from peers
   where exten="..exten
4  --
5  local result = channel["ODBC_SQL(".. 
   QueryString..")"]:get()
6  --
```

pbx_lua: Вызов системных команд

С помощью библиотеки ***io*** можно вызвать системную команду:

```
--
local handle=io.popen("/bin/hostname -s")
local hostname = handle:read("*a")
channel.__SERVERNAME:set(string.gsub(hostname, "\n", ""))
handle:close()
app.NoOp("[main]: Server name is >>>"..channel.SERVERNAME:get().."<<<")
--
```

pbx_lua: Ввод данных от пользователя

pbx_lua: Ввод данных от пользователя

- app.Read()

```
Read(variable,filename&[filename2[&...]],[maxdigits,[options,[attempts,[timeout]]]])
```

pbx_lua: Ввод данных от пользователя

- app.Read()

```
Read(variable,filename&[filename2[&...]],[maxdigits,[options,[attempts,[timeout]]]])
```

- app.BackGround() + app.WaitExten()

```
BackGround(filename1&[filename2[&...]],[options,[langoverride,[context]]])

WaitExten([seconds,[options]])
```

AC '22

pbx_lua: Ввод данных от пользователя

app.BackGround() + app.WaitExten()

AC '22

pbx_lua: Ввод данных от пользователя

app.BackGround() +
app.WaitExten()

```
1   --
2   -- Пример простого dialplan
3   --
4   function main( context, extension )
5       --
6       app.Answer()
7       --
8       app.Playback("tt-monkeys")
9       --
10      app.Hangup()
11      --
12  end
13  --
14  extensions = {
15      ["default"] = {
16          ["_[a-Z0-9]."] = main;
17          ["h"] = function () app.Hangup(16) end;
18      };
19  }
```

AC '22

pbx_lua: Ввод данных от пользователя

app.BackGround() + app.WaitExten()

```
app.Goto("ivr,s,1")
```

pbx_lua: Ввод данных от пользователя


```
1   extensions = {
2       ["default"] = {
3           ["_[a-Z0-9]."] = main;
4           ["h"] = function () app.Hangup(16) end;
5       };
6       ["ivr"] ={
7           ["s"]           = ivrStart;
8           ["_[0-9*#]"]    = ivrChoice;
9           ["i"]           = ivrInvalid;
10          ["t"]           = ivrTimeoute;
11          ["h"]           = function () app.Hangup(16) end;
12      }
13  }
```

AC '22

pbx_lua: Совет

При использовании в Lua циклов и app.Read() – проверяйте канал на Hangup

```
if tonumber(channel["CHANNEL(checkhangup)"]:get())==1 then
    app.Hangup()
end
```

pbx_lua: Отладка

pbx_lua: Отладка

- Dev сервер

AC '22

pbx_lua: Отладка

- Dev сервер
- Локальная на компьютере разработчика

pbx_lua: Отладка

- Dev сервер
- Локальная на компьютере разработчика

(необходимо объявить таблицы ***app*** и ***channel*** и заполнить их функциями-заглушками)

AC '22

Ссылки для изучения

«Родной» extensions.lua из исходников Asterisk
https://github.com/asterisk/asterisk/blob/master/configs/samples/extensions.lua.sample

ссылка на доклад Matthew Nicholson
https://clck.ru/325Law

AC '22

Ссылки для изучения

- Диалплан на LUA для Asterisk (Хабр) от Sayman_nsk
- Asterisk + LUA: быстрый старт (Хабр) от Antirek (Сергей Дмитриев)

И другие статьи этих уважаемых авторов

AC '22

Примеры extensions.lua для изучения (80 lvl ☺)

Статьи Юрия Горличенко на Хабр’е:

- Asterisk. Ненормальный перевод
- Asterisk Manager Interface в диалплане

AC ‘22

Спасибо за внимание!

Ссылка на презентацию этого доклада

У кого-нибудь есть вопросы?

spa79@bk.ru
Telegram @spa79

Пожалуйста, дайте оценку моему докладу прямо сейчас

AC '22