**Частное техническое задание на создание информационной системы запуска, отображения состояния и управления программными модулями, объединяемыми в одно автоматизированное рабочее место**

**Модуль: А-Интегратор**

Исполнитель: Фаткулин Д.А.

Руководитель: Рязанов В.С.

Заказчик: Ханов Э.Б.

**1. АННОТАЦИЯ**

ЧТЗ на создание автоматизированной системы, предназначенной для запуска, останова, отображения состояния и управления модулями, входящими в состав АРМ, с единого графического пользовательского интерфейса.

Система узнает о составе и возможностях отображения состояния запускаемых модулей из конфигурационных файлов, предоставляемых разработчиками программных модулей согласно формату, описанному в настоящем документе.

Обмен статусной и управляющей информацией с управляемыми модулями производится с использованием TCP-сокетов на основе протокола обмена, использующего Google Protocol Buffers. Протокол обмена описывается в настоящем документе.

ГИП Интегратора описывается в дизайн-проекте, являющимся приложением к настоящему ЧТЗ.

**2. СПИСОК СОГЛАСОВАНИЯ**

**3. СПИСОК СОКРАЩЕНИЙ**

|  |  |
| --- | --- |
| АРМ | Автоматизированное рабочее место |
| АС | Автоматизированная система |
| ГИП | Графический интерфейс пользователя |
| ЗМ | Запускаемый модуль |
| ПО | Программное обеспечение |
| ТЗ | Техническое задание |
| ЧТЗ | Частное техническое задание |
|  |  |

**4. СПИСОК ТЕРМИНОВ**

**Запускаемый модуль** - программный комплекс, входящий в состав АРМ, который должен запускаться в составе единого ГИП и спроектирован с учетом требований по интеграции.

**Конфигурационная директория** - каталог в файловой системе, предназначенный для хранения конфигурационных файлов запускаемых модулей.

**Представление** - область на ГИП Интегратора, предназначенная для отображения состояния ЗМ и размещения элементов управления ЗМ.

**5. НАЗНАЧЕНИЕ И ЦЕЛИ СОЗДАНИЯ СИСТЕМЫ**

**5.1. Назначение системы**

Интегратор предназначен для объединения в единый ГИП различного ПО, входящего в единый АРМ.

**5.2. Цели создания системы**

**6. ТРЕБОВАНИЯ К СИСТЕМЕ**

**6.1. Требования к системе в целом**

АС должна обеспечивать:

* возможность запуска процессов программных модулей, входящих в состав единого АРМ,
* возможность создания нескольких АРМ, в которые могут входить разные ЗМ,
* работу с одним активным АРМ,
* получение статусной информации и отправку управляющих сигналов всем ЗМ, входящим в состав АРМ

Система должна запускаться под управлением операционных систем Windows 7, Windows 8, Windows 10, а также под управлением Astra Linux 1.3

**6.2. Требования к конфигурированию АС**

Конфигурирование АС производится с помощью специальных конфигурационных файлов в XML-формате. Эти файлы расположены в конфигурационной директории, расположение которой зависит от типа операционной системы:

Для ОС семейства Windows конфигурационная директория расположена в

c:\PostWin Software\AIntegrator\Config\

Для ОС семейства Linux конфигурационная директория расположена в

/etc/default/AIntegrator/Config

**6.2.1. Конфигурационный файл AIntegrator.xml**

Содержит информацию о настройках Интегратора, количестве и составе АРМ, входящих в его конфигурацию, признаки группировки ЗМ в составе АРМ.

Файл AIntegrator.xml имеет следующий формат:

<a\_integrator ip\_address="127.0.0.1" port="50200">

<work\_station name="АРМ-1">

<module\_group name="Группа-1">

<execution\_module id="Модуль-1"/>

<execution\_module id="Модуль-2"/>

.

.

</module\_group>

.

.

</work\_station>

.

.

</a\_integrator>

Структура описывается в виде следующих XML-тегов:

1. Тег <a\_integrator> является внешним контейнером документа и содержит внутри себя описание структуры АРМ. Содержит обязательные атрибуты ip\_address для указания TCP-адреса Интегратора (пока не используется, т.к. Интегратор всегда запускается локально) и port для указания номера TCP-порта, который прослушивает Интегратор для установления входящих соединений.
2. Теги <work\_station> описывают АРМ, входящие в состав Интегратора. Содержит атрибут name, отражающий название АРМ.
3. Теги <module\_group> предназначены для группировки ЗМ в составе одного АРМ. Атрибут name описывает название группы. Если группа одна, то она может не иметь атрибута name.
4. Теги <execution\_module> описывают запускаемые модули, входящие в состав АРМ. Имеют единственный атрибут id, который служит для связывания с параметрами ЗМ, которые описываются в конфигурационном файле запускаемого модуля.

**6.2.2. Конфигурационный файл запускаемого модуля**

Все файлы, расположенные в конфигурационной директории и имеющие расширение xml, кроме перечисленных в п. 6.2.1, считаются файлами конфигурации запускаемых модулей. Они должны содержать следующую информацию о ЗМ:

* название,
* наименование АРМ, в которые входит данный ЗМ,
* путь в файловой системе к исполняемому файлу запускаемого процесса, рабочую директорию выполняемого процесса, а также список параметров, необходимых для запуска,
* дополнительную информацию для отображения на ГИП Интегратора

Файл конфигурации ЗМ имеет следующий формат:

<execution\_module version="1" name="Моё приложение" id="my1">

<timing\_parameters

module\_launch\_delay\_sec="20"

status\_update\_sec="5"

launch\_interval\_sec="2"/>

<launch\_list>

<launch\_parameters

path\_to\_executable="/opt/myapp/executable"

command\_options="--do\_some\_think --debug=true"

work\_directory="/opt/myapp"/>

.

.

</launch\_list>

<card\_list>

<card id="1" type="simple" background\_color="red"/>

.

.

</card\_list>

<work\_mode\_list>

<work\_mode id="jam" name="Подавление"/>

<work\_mode id="ree" name="Оценка РЭО"/>

.

.

</work\_mode\_list>

<status\_file\_list>

<status\_file id="1" path="/usr/share/icons/my\_cool\_icon.png"/>

<status\_file id="2" path="/usr/share/icons/other\_cool\_icon.png"/>

.

.

</status\_file\_list>

</execution\_module>

Структура описывается в виде следующих XML-тегов:

* Тег <execution\_module> является внешним контейнером документа и содержит внутри себя описание параметров запускаемого модуля. Содержит атрибуты version, для указания версии протокола конфигурационного файла ЗМ, name для описания названия модуля, id для определения идентификатора ЗМ, который должен соответствовать описанию в файле Интегратора из п. 6.2.1
* Тег <timing\_parameters> описывает параметры временнЫх интервалов. Атрибут module\_launch\_delay\_sec содержит время, которое отводится модулю на запуск и на установление соединения с Интегратором. Если по истечении module\_launch\_delay\_sec соединение не установлено, то считается, что запуск модуля совершен неудачно, о чем сообщается оператору. Атрибут status\_update\_sec содержит интервал обновления статуса модуля. Если по истечении status\_update\_sec статус не получен, считается, что модуль работает некорректно. Т.е. для корректной работы с Интегратором, запускаемому модулю необходимо обновлять статус не реже чем status\_update\_sec, даже если изменений не было. Атрибут launch\_interval\_sec содержит интервал времени, который должен пройти между запуском процессов, указаных в праметрах ЗМ.
* Тег <launch\_list> является контейнером для указания параметров запуска модуля. Атрибутов не имеет. Процессы запускаются последовательно в порядке перечисления в списке.
* Теги <launch\_parameters> содержит информацию о параметрах запуска процессов. Атрибут path\_to\_executable содержит путь к запускаемому файлу, атрибут command\_options содержит строку параметров, передаваемых процессу при его старте, атрибут work\_directory содержит рабочую директорию для запуска процесса.
* Тег <card\_list> является контейнером для указания списка карточек поста, которые будут отображаться на ГИП Интегратора. Атрибутов не имеет.
* Теги <card> описывают карточки поста для отображения на ГИП Интегратора. Атрибут id содержит идентификатор карточки. Атрибут type содержит тип карточки (поддерживаемые значения: "simple"). Атрибут background\_color описывает цвет фона карточки на ГИП Интегратора (поддерживаемые значения: "red", "green", "blue", "lightblue", "yellow", "purple").
* Тег <work\_mode\_list> является контейнером для указания списка режимов работы. Используется для отображения списка режимов поста на ГИП Интегратора. Если список пуст, то на ГИП Интегратора режимы работы не отображаются. Атрибутов не имеет.
* Теги <work\_mode> описывают режимы работы поста. Атрибут id содержит идентификатор режима работы. Атрибут name содержит название режима работы для отображения на ГИП Интегратора.
* Тег <status\_file\_list> является контейнером для указания списка настраиваемых статусных иконок, которые могут быть использованы запускаемым модулем для отображения статусов на ГИП Интегратора. Атрибутов не имеет.
* Теги <status\_file> регистрируют файлы иконок. Атрибут id описывает идентификатор иконки. Атрибут path содержит путь до графического файла иконки в файловой системе.

**6.3. Требования к обмену информацией**

АС должна обеспечивать возможность обмена статусной информацией и управляющими воздействиями между Интегратором и запускаемыми модулями. Обмен реализуется с использованием TCP-сокетов на транспортном протоколе TCP/IP. На прикладном уровне сообщения формируются и передаются с использованием Google Protocol Buffer версии 2.

С точки зрения клиент-серверной архитектуры, Интегратор выступает в качестве сервера, который при запуске открывает TCP-сокет и начинает прослушивать TCP-порт, указанный в настройках п. 6.2.1. Соединение инициализирует ЗМ, который выступает в качестве клиента и обеспечивает сохранение соединения.

**6.3.1. Описание логического уровня протокола**

Сообщения в зависимости от типа могут передаваться в обоих направлениях. Инициатор обмена отправляет сообщение-запрос, в котором помимо прочей информации содержится идентификатор сообщения. Получатель обрабатывает сообщение и формирует сообщение-ответ, в котором помимо прочего содержится идентификатор сообщения (соответствующий запросу) и отчет о выполнении.

Идентификатор сообщения используется для однозначной связи запрос-ответ и должен инкрементироваться перед отправкой очередного запроса.

Отчет о выполнении содержит информацию об успешности обработки запроса и, в случае возникновения ошибки, описание проблемы, возникшей при обработке.

Схема взаимодействия:



**6.3.2. Описание пакетного уровня протокола**

После установки TCP-соединения между клиентом и сервером между ними образуется канал приема-передачи потока байт. Помещение в этот поток пакетов на стороне передачи и разделение потока на пакеты на стороне приема осуществляется следующим образом:

Каждый пакет имеет структуру:

|  |  |
| --- | --- |
| 4 байта. Размер информационного пакета  (В формате Little Endian) | Информационный пакет.  Сериализованное protobuf-сообщение |

Первые четыре байта определяют длину информационного сообщения, после которых идет сериализованное сообщение Google Protocol Buffer.

**6.3.2. Описание прикладного уровня протокола**

Все типы protobuf-сообщений, использующиеся в протоколе содержатся в пространстве имен ai.proto с использованием директивы:

package ai.proto;

Названия пользовательских типов пишутся в UpperCamelCase, в том числе и сокращения. Имена полей пишутся в lower\_snake\_case, в том числе и сокращения.

Все поля сообщений, имеющие тип string, передаются в кодировке UTF-8.

**6.3.3. Структура базового сообщения**

Базовое сообщение протокола обмена имеет следующее описание:

message AiMessage {

enum Type {

HANDSHAKE\_REQUEST = 10;

HANDSHAKE\_RESPONSE = 11; SET\_PARAMETERS\_REQUEST = 20; SET\_PARAMETERS\_RESPONSE = 21 SET\_STATE\_REQUEST = 30; SET\_STATE\_RESPONSE = 31; SET\_WORK\_MODE\_REQUEST = 40; SET\_WORK\_MODE\_RESPONSE = 41; CONTROL\_ACTION\_REQUEST = 50; CONTROL\_ACTION\_RESPONSE = 51;

}

required Type type = 1;

required uint32 id = 2;

optional HandshakeRequest handshake\_request = 10;

optional HandshakeResponse handshake\_response = 11;

optional SetParametersRequest set\_parameters\_request = 20;

optional SetParametersResponse set\_parameters\_response = 21;

optional SetPostStateRequest set\_state\_request = 30;

optional SetPostStateResponse set\_state\_response = 31;

optional SetWorkModeRequest set\_work\_mode\_request = 40;

optional SetWorkModeResponse set\_work\_mode\_response = 41;

optional ControlActionRequest control\_action\_request = 50;

optional ControlActionResponse control\_action\_response = 51;

}

Обязательными полями являются type, содержащее тип передаваемого сообщения и id, хранящее идентификатор сообщения. Опциональные поля заполняются в зависимости от типа и передают содержательную часть сообщения.

**6.3.4. Установление соединения. Сообщение HANDSHAKE**

При установлении соединения, запускаемый модуль первым присылает сообщение HANDSHAKE\_REQUEST, которое имеет следующее описание:

message HandshakeRequest {

required string host\_id = 1;

required SoftwareVersion host\_version = 2;

}

Оно содержит идентификатор запускаемого модуля и версию его ПО. В ответ от Интегратора приходит сообщение типа HANDSHAKE\_RESPONSE, имеющее вид:

message HandshakeResponse {

required Report report = 1;

required SoftwareVersion integrator\_version = 2;

}

В ответе приходит отчет о выполнении и версия ПО Интегратора. Отчет о выполнении может содержать предупреждение о несоответствии версий протоколов взаимодействия. Версия ПО содержится в сообщении следующего вида:

message SoftwareVersion {

enum ProtocolCurrentVersion {

CURRENT = 1;

}

enum ProtocolSupportedVersion {

MIN = 1;

}

required uint32 protocol\_version\_current = 1;

required uint32 protocol\_version\_min\_supported = 2;

required string software\_version = 11;

required string build\_date\_time = 12;

}

**6.3.5. Установка параметров. Сообщение SET\_PARAMETERS**

Установка параметров поста производится сообщением типа SET\_PARAMETERS\_REQUEST

**6.3.6. Установка состояния. Сообщение SET\_STATE**

Установка состояния производится базовым сообщением типа SET\_STATE\_REQUEST, с описанием:

message SetPostStateRequest {

required PostState post\_state = 1;

}

где:

message PostState {

required string card\_id = 1;

required StateType type = 2;

optional SimpleState simple\_state = 3;

}

enum StateType {

ST\_UNKNOWN = 1;

ST\_SIMPLE = 2;

}

Сообщение PostState содержит описание состояния поста и состоит из следующих полей:

* card\_id - Идентификатор карточки поста, на которую будет выводиться статусная информация. Настраивается в конфигурационном файле запускаемого модуля
* type - Тип состояния, соответствующий карточке отображения
* simple\_state - Описание состояния "простой" карточки. Поле должно быть заполнено, если тип карты равен ST\_SIMPLE

Состояние текущего режима работы описывается следующим перечислением:

enum PostModeStatus {

PM\_IDLE = 1; // Пост остановлен.

PM\_STARTING = 2; // Пост запускается для работы в заданном режиме

PM\_RUN = 3; // Пост работает в указанном режиме

PM\_STOPPING = 4; // Пост останавливает работу в указанном режиме

}

Состояние "простой" карточки описывается следующим образом:

// Состояние поста, соответствующее простой стандартной карточке

message SimpleState {

required StatusIcon main\_status = 1;

repeated StatusIcon additional\_statuses = 2

repeated string text\_statuses = 3;

optional string current\_mode\_id = 4;

optional PostModeStatus mode\_status = 5;

}

Сообщение SimpleState состоит из следующих полей:

* main\_status - Основной статус
* additional\_statuses - Дополнительные статусы, количество [0..5]
* text\_statuses - Текстовые статусы, количество [0..3]
* current\_mode\_id - Индекс текущего установленного режима работы. Должен быть установлен, если в конфигурационном файле задано более одного режима
* mode\_status - Статус режима работы

Отображение состояния "простой" карточки отражено на следующем рисунке:

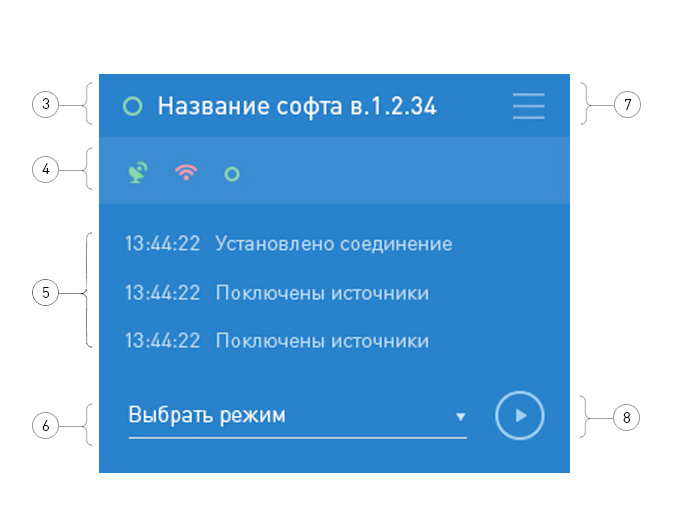


Рис. 1. Отображение статусной информации на "простой" карточке

В поле 3 отображаются основной статус и название запускаемого модуля. В поле 4 показываются дополнительные статусы в виде графических иконок (до пяти статусов). В поле 5 отображаются текстовые статусы, которые служат для показа текстовой информации. В поле 6 показывается текущий режим работы поста с возможностью выбора.

Статусные иконки передаются в виде следующего сообщения:

message StatusIcon {

required StatusIconType type = 1;

required string description = 2;

optional StandardIcon standard\_icon = 3;

optional CustomIcon custom\_icon = 4;

}

Поле description содержит текстовое описание статуса. Данное описание отображается при наведении курсора мыши на иконке статуса на ГИП Интегратора.

Иконки могут быть двух типов:

enum StatusIconType {

SIT\_STANDARD = 1;

SIT\_CUSTOM = 2;

}

Стандартные иконки входят в состав ПО Интегратора и могут быть использованы без дополнительной установки. Стандартная иконка описывается следующей структурой:

message StandardIcon {

required IconColor color = 1;

required IconType type = 2;

}

Соотношение типа иконки и ее графического представления показано в Приложении № 1

Также в качестве статусной иконки можно указать идентификатор зарегистрированного файла изображения в конфигурационном файле ЗМ. Это делается с помощью следующего сообщения:

message CustomIcon {

required string custom\_file\_id = 1;

}

**6.3.7. Установка режима работы. Сообщение SET\_WORK\_MODE**

Установка состояния производится базовым сообщением типа SET\_WORK\_MODE\_REQUEST, со следующим дополнительным описанием:

message SetWorkModeRequest {

required string work\_mode\_id = 1;

optional string card\_id = 2;

}

* Поле work\_mode\_id содержит идентификатор режима работы, который нужно установить
* Поле card\_id содержит идентификатор карты (Если отправляется от Интегратора, то означает id карты, с которой отправлена команда. Если отправляется от запускаемого модуля, то означает на какой карточке надо установить режим, если не задано, режим устанавливается на всех карточках)

В ответ приходит базовое сообщение типа SET\_WORK\_MODE\_RESPONSE, с дополнительным описанием вида

message SetWorkModeResponse {

required string work\_mode\_id = 1;

required Report report = 2;

}

где:

* work\_mode\_id - Установленный режим работы. Если запрос выполнен без ошибок, должен быть равен work\_mode\_id из запроса
* report - Отчет о выполении

**6.3.7. Управляющие воздействия. Сообщение CONTROL\_ACTION**

Отправка управляющих воздействий производится базовым сообщением типа CONTROL\_ACTION\_REQUEST, со следующим дополнительным описанием:

message ControlActionRequest {

required ControlAction action = 1; // Управляющее воздействие

}

где

message ControlAction {

enum Type {

CA\_KILL\_PROGRAM = 1; // Завершить работу модуля

CA\_RAISE\_WINDOW = 2; // Показать рабочее окно модуля поверх всех окон

CA\_START\_WORK = 3; // Запустить работу модуля

CA\_STOP\_WORK = 4; // Остановить работу модуля

}

required Type type = 1; // Тип управляющего воздействия

required string card\_id = 2; // Идентификатор карты, с которой было отправлено управляющее воздействие

}

Управляющие воздействия отправляются Интегратором в ответ на действия оператора. Обработка и интерпритация данных воздействий возлагается на разработчиков запускаемых модулей. Но для универсализации взаимодействия с Интегратором предполагаются следующая реакция на команды:

* CA\_KILL\_PROGRAM - при получении данного воздействия, запускаемый модуль должен произвести корректное завершение программы с сохранением результатов работы
* CA\_RAISE\_WINDOW - запускаемый модуль должен показать окно своего приложения поверх всех остальных окон. В зависимости от card\_id или текущего режима работы могут быть показаны различные окна.
* CA\_START\_WORK - запустить работу поста в соответствии с выбранным режимом работы
* CA\_STOP\_WORK - остановить работу поста

**Приложение 1. Стандартные статусные иконки**

Ниже представлена таблица соответствия типа иконки и ее графического представления.

|  |  |  |  |
| --- | --- | --- | --- |
| E:\tmp\1111\image\brightness_1.png | IT\_CIRCLE | E:\tmp\1111\image\photo_camera.png | IT\_CAMERA |
| E:\tmp\1111\toggle\radio_button_unchecked.png | IT\_RING | E:\tmp\1111\action\explore.png | IT\_DF |
| E:\tmp\1111\maps\beenhere.png | IT\_RHOMBUS | E:\tmp\1111\hardware\memory.png | IT\_HARDWARE |
| E:\tmp\1111\toggle\check_box_outline_blank.png | IT\_SQUARE | E:\tmp\1111\maps\place.png | IT\_LOCATION |
| E:\tmp\1111\action\change_history.png | IT\_TRIANGLE | E:\tmp\1111\maps\map.png | IT\_MAP |
| E:\tmp\1111\av\fast_rewind.png | IT\_BACKWARD | E:\share.png | IT\_NETWORK |
| E:\tmp\1111\av\fast_forward.png | IT\_FORWARD | E:\tmp\1111\action\settings_input_antenna.png | IT\_RADAR |
| E:\tmp\1111\av\play_arrow.png | IT\_PLAY | E:\tmp\1111\notification\wifi.png | IT\_WIRELESS |
| E:\tmp\1111\av\pause.png | IT\_PAUSE | E:\tmp\1111\av\equalizer.png | IT\_SPECTRUM |
| E:\tmp\1111\av\stop.png | IT\_STOP | E:\tmp\1111\communication\call.png | IT\_CALL |
| E:\tmp\1111\social\people.png | IT\_GROUP | E:\call_end.png | IT\_CALL\_END |
| E:\tmp\1111\social\person.png | IT\_OPERATOR | E:\tmp\1111\communication\message.png | IT\_SMS |
| E:\tmp\1111\communication\contacts.png | IT\_USER | E:\tmp\1111\action\info.png | IT\_INFO |
| E:\tmp\1111\file\cloud.png | IT\_CLOUD | E:\tmp\1111\alert\warning.png | IT\_WARNING |
| E:\tmp\1111\hardware\computer.png | IT\_COMPUTER | E:\tmp\1111\alert\error.png | IT\_ERROR |
| E:\tmp\1111\device\storage.png | IT\_DATABASE | E:\tmp\1111\content\report.png | IT\_REPORT |
| E:\tmp\1111\file\folder.png | IT\_DIRECTORY | E:\tmp\1111\content\flag.png | IT\_FLAG |
| E:\tmp\1111\action\description.png | IT\_FILE | E:\tmp\1111\content\block.png | IT\_BLOCK |
| E:\tmp\1111\content\link.png | IT\_LINK | E:\tmp\1111\hardware\speaker.png | IT\_SPEAKER |
| E:\tmp\1111\notification\sd_card.png | IT\_MEMORYCARD |  |  |
| E:\tmp\1111\hardware\desktop_windows.png | IT\_MONITOR |  |  |
| E:\tmp\1111\action\print.png | IT\_PRINTER |  |  |
| E:\tmp\1111\social\public.png | IT\_WEB |  |  |
| E:\tmp\1111\av\mic.png | IT\_MICROPHONE |  |  |