

1. 06. Installing .....	2
2. 12. Workshop: autotests' workflow .....	4
2.1 A. Workshop: installing .....	5
2.2 B. Workshop: injection .....	6
2.3 C. Workshop: tests .....	10
2.3.1 1. Configuring .....	11
2.3.2 2. Running the tests .....	13
2.3.3 3. CI (optional) .....	14
2.3.4 4. Drill Web App .....	15

## 06. Installing

[О конфигах.](#)

### Зависимости

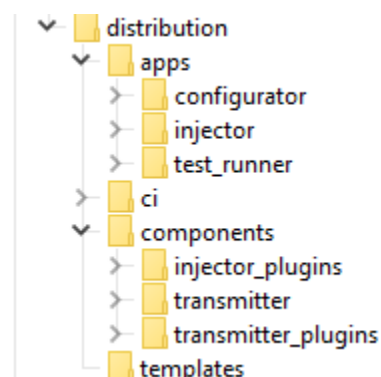
1. SDK для [.NET 6](#), а также для [.NET 5](#) при использовании *Agent.Standard.Tester* (внутреннее тестирование QA). Уже установленные фреймворки можно узнать консольной командой: `dotnet --info`.
2. Net Framework Developer Pack версии [4.8](#).
3. Скачанный с [сайта](#) или переданный пользователю иным способом файл дистрибуции *docker-compose-run-distribution.yml*.

[Continuous Integration \(CI\) with .NET SDK and tools - .NET CLI | Microsoft Docs](#)

Чтобы получить дистрибутив подпрограмм *Drill4Net*, необходимо использовать файл дистрибуции:

1. Положить *docker-compose-run-distribution.yml* в директорию, где будет располагаться папка дистрибутива.
2. Удостовериться, что в файле установлена необходимая версия дистрибутива: *image: ghcr.io/epamx/drill4net-win-distribution:v0.9.0-195*
3. Выполнить команду в *cmd* или *PowerShell*: `docker-compose -f docker-compose-run-distribution.yml up`
4. После выполнения команды рядом с файлом появится папка дистрибутива *distribution*. Её можно соответствующе переименовать.

Структура папок дистрибутива:



В папке *apps* располагаются запускаемые пользователем программы, а в *components* - плагины инъектора и компоненты трансмиттера, внедряемого в исследуемый таргет. В корне лежит *compose-файл* для установки инфраструктуры *Drill* в контейнерах Docker и PDF-документация: краткий практикум с примером автотестов для проекта *IHS Markit*, а также общая версия документации (находящаяся в постоянной разработке). В папке *templates* - различные модельные конфиги, используемые Конфигуратором для формирования соответствующих пользовательских. Папка *ci* пустая - по умолчанию она используется Конфигуратором для хранения настроек *CI*.

### Установка сервисов в Docker

Далее, если нужно установить локальные сервисы в Docker, необходимо использовать файл *docker-compose.yml* из корня дистрибутива, выполнив в *cmd* или *PowerShell* команду:

```
docker-compose up -d
```

Будут установлены:

1. *Drill admin* (service + web app). Веб-приложение - <http://localhost:8091/>. Дополнительно имеется WebAPI, Swagger - <http://localhost:8090/apidocs>.
2. *Agent Server*.
3. Сервер сообщений *Kafka*. Адрес по умолчанию: *localhost:9093*. Все его топик, касающиеся *Drill4Net*, имеют префикс *d4n-*. Аутентификация будет реализована позже при необходимости.
4. *ZooKeeper* - сервис, контролирующий работоспособность *Kafka*.
5. *Kafdrop* - администраторское веб-приложение для *Kafka* - при необходимости может использоваться для технического контроля. Адрес подключения по умолчанию: <http://localhost:9000/>.

### Проверка результата


Секунд через 5 после запуска контейнеров можно зайти в веб-браузере по адресу <http://localhost:8091/> - должна быть видна *админка Drill* (список таргетов-агентов пока пуст).

 По адресу <http://localhost:8090/apidocs/index.html> находится Swagger REST-сервиса, а по <http://localhost:9000/> - консоль управления сервером *Kafka*, однако пользователям для определения *test coverage* они не нужны.

В Докере также запускается *Agent Service* - как *drill4net-service*. Его логи выглядят примерно так (и в том числе сохраняются в локальной папке *logs\_drill*):

```
2022-02-02 15:36:39.918|Information|GetServersFromEnv|The environment variable for message server addresses is empty. - will be used the config's value
2022-02-02 15:36:39.952|Debug|AgentServer|ServerHost|.ctor|ServerHost created: Drill4Net.Agent.Service 0.9.40.34-main+47ea925e
2022-02-02 15:36:39.976|Information|AgentServer|AgentServer|.ctor|Worker path: [D:\Projects\EPM-D4J\Drill4Net\build\bin\Debug\Drill4Net.Agent.Worker\net6.0\Drill4Net.Agent.Worker.exe]
2022-02-02 15:36:39.977|Debug|AgentServer|AgentServer|.ctor|Created.
2022-02-02 15:36:39.977|Information|AgentServer|ServerHost|ExecuteAsync|ServerHost ready.
2022-02-02 15:36:40.418|Debug|AgentServer|PingKafkaReceiver`1|Start|Start.
2022-02-02 15:36:40.418|Debug|AgentServer|TargetInfoKafkaReceiver`1|Start|Start.
2022-02-02 15:36:40.468|Debug|AgentServer|PingKafkaReceiver`1|RetrievePings|Start retrieving pings...
2022-02-02 15:36:40.469|Information|AgentServer|TargetInfoKafkaReceiver`1|RetrieveTargets|Start retrieving target info...
2022-02-02 15:36:40.470|Debug|AgentServer|TargetInfoKafkaReceiver`1|RetrieveTargets|Target info servers: localhost:9093
2022-02-02 15:36:40.471|Debug|AgentServer|TargetInfoKafkaReceiver`1|RetrieveTargets|Target info topics: d4n-server-target-info
```

Установка и запуск *сервисов* окончены.

 Практикум на примере дистрибутива для проекта IHS см. в [следующем разделе](#).

## Замечания по дистрибутиву

Следует пользоваться готовой структурой дистрибутива, а создание и проверку конфигов производить с помощью соответствующих команд [Конфигуратора](#). Однако *если* дистрибутив с *готовой* структурой папок и сконфигурированными путями к компонентам системы не передан, проследить за связями в конфигах придётся самостоятельно. Необходимые компоненты:

1. Собственно иньектор. В папку *injector* разворачиваем дистрибутив Иньектора. Полное название программы - *Drill4Net.Injector.App*. Инсталлятора нет - достаточно запустить exe-файл. Для каждого конкретного использования необходимо использовать конфиг, [настраиваемый](#) отдельно для каждого *исследуемого приложения* (таргета). Конфиги могут использоваться программой автоматически, а можно передать путь к конкретному в командной строке.
2. Плагины для Иньектора. Расположить в *components/injector\_plugins* (вариант: *plugins* в директории Иньектора), и указать к ней путь в конфиге иньектора.
3. Компонент Трансмисмиттер разворачиваем в папку *components/transmitter*. Имя конфига по умолчанию - *svc.yml*, при необходимости изменить *нестандартные* ноды сервера Кафки (свойство *Servers*). Аутентификация будет реализована позже.
4. Плагины Трансмисмиттера - в отдельную папку, например, *transmitter\_plugins*. В конфиге *Иньектора* (свойство *Plugins*) указываются пути для каждого такого плагина в отдельности (далее там появятся и другие параметры по их внедрению/запуску). Разворачиваем все имеющиеся папки плагинов *Трансмисмиттера* (не путаем с плагинами Иньектора, которые будут впоследствии). Каждый плагин имеет свой *собственный* специфический конфиг, но как правило, его не нужно трогать.
5. Программа *Test Runner*. В принципе, может лежать где угодно.
6. Программа *Configurator (d4n.exe)* - впоследствии название может быть изменено. В принципе, может лежать где угодно. Использует конфиг *app.yml*, в котором прописываются пути к остальным программам и некоторые другие параметры.

Также необходимо отметить:

1. Папка для *Transmitter* должна располагаться в том окружении (host, докер), в котором будет выполняться инструментированный *таргет*, с правом чтения.
2. В конфигах по возможности лучше использовать относительные пути, при этом *полные* пути рассчитываются относительно *запущенной программы*, а не расположения конфига.
3. Перед использованием системы рекомендуется запустить программу Конфигуратор и настроить общие параметры *Drill4Net*, а также его собственные. Для этого пригодятся команды *"sys cfg"* и *"sys restore"*. Настройки собственно Конфигуратора хранятся в локальном конфиге *app.yml*, однако системные настройки в конечном итоге переносятся в ряд *отдельных* конфигов, включая *app.yml* программы Иньектор.

## 12. Workshop: autotests' workflow

Для .NET воркфлоу определения *test coverage* состоит из двух стадий: оффлайнной обработки анализируемого проекта (таргета, или SUT - *system under test*) и собственно сбора информации во время запуска последнего. В общем случае следует:

1. Обеспечить настройку и запуск инфраструктуры Drill4Net (*Drill Admin/UI, Agent Server, Kafka*).
2. В оффлайне провести инъекцию исходных сборок исследуемого проекта программой *Инъектор*.
3. Запустить инструментированное приложение или *Test Runner* в случае автотестов. Конфигурирование таргета в его *собственных* настройках не требуются.
4. Зарегистрировать нового агента в веб-приложении *Drill* (*админке*).
5. Получить данные и проанализировать их.

Конфигурирование на всех стадиях а также их запуск можно проводить как полностью вручную, так и использовать [Configurator](#).

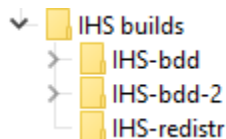
Стадии практикума:

# A. Workshop: installing

Реквизиты:

1. Операционная система Windows 10, [SDK .NET 6](#), [SDK .NET Core 3.1](#). Уже установленные фреймворки можно узнать консольной командой: `dotnet --info`.
2. Установленный [Docker Desktop](#).
3. Скачанный с [сайта](#) или переданный пользователю иным способом файл дистрибуции `docker-compose-run-distribution.yml`.
4. Два билда проекта *IHS Markit Investor Access* для BDD-тестов (*Ipreno.Csp.IaDeal.Api.Bdd.Tests*). Во втором из них изменён метод `ContainsProperties` класса `ILMapperExtensions`. Распаковать архивы в отдельные папки *как можно ближе к корню диска* (связано со сторонней зависимостью DacPac, который использует Ruby, а последний не умеет работать с очень длинными путями файлов).

Структура папок распакованных билдов (как пример):



Она содержит собственно скомпилированные "чистые" билды и специальную папку *IHS-redistr*, необходимую для запуска тестов *не* на компьютерах разработчиков.

Получение дистрибутива Drill4Net и установку сервисов [провести](#) обычным образом.

## B. Workshop: injection

Для прохождения практикума лучше всего иметь в наличии как минимум два билда одного и того же проекта. Для первого из них будет проведён первичный сбор информации о классах и покрытии кода тестами, а на последующих - показаны бенефиты от использования системы *Drill* - в данном случае, автотестировании на проекте *IHS Markit (Investor Access)*. Билды должны различаться каким-либо функциональным изменением в *product*-классах (не тестовых).

Предположим, мы расположили два "чистых", исходных билда в двух разных папках (распаковать, если они в архивах):

```
d:\Projects\IHS-bdd\  
d:\Projects\IHS-bdd-2\  

```

Далее есть два варианта: высокоровневый, более простой способ с использованием программы *Configurator*, и более сложный, низкоуровневый прямого редактирования конфигурационных файлов. Для последнего случая, в рамках данного Workshop, будут использованы почти готовые конфиги, однако всё равно будут требоваться дополнительные действия.

### Вариант А: Конфигуратор

Необходимо запустить программу [Конфигуратор](#) - *d4n.exe* из папки *apps\configurator*, ввести команду "*new trg*" (без кавычек) и ответить на несколько вопросов. Всё, чем отличаются ответы для двух билдов для данного workshop, это место расположения папок с исходными, "чистыми" файлами и, соответственно, их версия (может быть условной).



В квадратных скобках после вопроса будут предложены варианты по умолчанию (но стоит их всё равно проверять). Чтобы принять его, достаточно просто нажать клавишу *Ввод*. Если ответ будет неверным или Конфигуратор не поймёт его, вопрос повторится. Чтобы выйти из "опросника" в главное меню программы, то есть, прекратить действие команды, достаточно ввести символ *q*.

**Q1:** *Target's directory (compiled assemblies). It can be full or relative (for the Injector program):*

**A1:** `d:\Projects\IHS-bdd\`

**Q2:** *Target's name (as Agent ID). It must be the same for the different builds of the Product: [ih-s-bdd]*

Здесь будет предложено имя агента, автоматически сформированное на основании папки таргета, но оно не всегда окажется удачным, и тогда лучше ввести своё. В любом случае, в них используются лишь буквы, цифры и дефисы.

**A2:** `IHS-bdd`

**Q3:** *Target's description:*

**A3:** `Injection for IHS target for Workshop (можно оставить пустым)`

**Q4:** *Type of retrieving the target build's version:*

(тут выведется несколько вариантов ответов в виде нумерованного списка)

Please make your choice: [1]

**A4:** (просто принимаем значение по умолчанию, нажав клавишу *Ввод*)

**Q5:** *Target's version (SemVer format):*

**A5:** `0.1.0 (для первого билда, а для второго - 0.2.0)`

Далее нужно будет указать, что именно инструментировать: какие файлы, классы, что принять во внимание, а что, возможно, пропустить. Это очень ответственный шаг и нужно понимать "внутренности" таргета. Программа выведет довольно большой блок информации, по сути, инструкцию, как это сделать. "Фильтр сущностей" формируется вводом строк специального формата по каждому типу сущности. Можно ввести отдельные строки по каждой из них, а можно - полную, склеенную строку. Когда с фильтром закончите, введите *ok*.

**Q6:** (тут будет много текста) *Please create at least one filter rule:*

**A6:** `NS=Ipreo;FILE=^Ipreo.Csp.IaDeal.Api.Bdd.Tests.dll,^Ipreo.Csp.IaDeal.Test.Infrastructure.dll;FLD=^*`

Здесь мы сообщили, что нужно инструментировать только классы с namespace, начинающимися с *Ipreo*, и нужно пропустить (символ циркумфлекса перед каждым значением) все внутренние папки, а также два файла (где бы они не находились): *Ipreo.Csp.IaDeal.Api.Bdd.Tests.dll* и *Ipreo.Csp.IaDeal.Test.Infrastructure.dll* - это сборки, содержащие тесты и их инфраструктуру. Однотипные сущности перечисляются через запятую, а сами типы разделяются точкой с запятой. Все эти части можно было бы ввести раздельно-последовательно.

Это один из редких случаев конфигурирования системы через программу, когда проще сделать то же самое, но редактируя соответствующий конфиг напрямую. Однако текущего конфига ещё нет.

**Q7:** *Configure destination directory for the instrumented target by:*

(далее нумерованный список вариантов)

*Please make your choice: [1]*

**A7:** (клавиша Ввод)

**Q8:** *Postfix to original directory: [Injected]*

**A8:** (клавиша Ввод)

**Q9:** *Does the target have any automated tests? [y]*

**A9:** (клавиша Ввод)

Теперь нужно указать *специфические* плагины, необходимые для инъекции правильных инструкций в таргет. На данный момент их автоподбора нет, но он, возможно, будет реализован позже. *IHS Investor Access* использует BDD-фреймворк *SpecFlow*. В стандартной поставке в именах таких плагинов будет постфикс *GeneratorContexter*.

**Q10:** *The found plugins are:*

1. *SpecFlowGeneratorContexter*

*Plugin number:*

**A10:** 1

У плагинов могут свои собственные конфиги (какого угодно содержания - это зависит от их логики). Далее просто следует указать его имя - по правилам "хорошего тона" ему стоит иметь префикс *plug\_* (конфигам собственно *инъекций* лучше добавлять префикс *inj\_*). Указанный ниже конфиг *plug\_IHS\_SpecFlow* должен существовать в папке Инъектора, и он должен быть настроен - однако из-за *потенциальной* специфики каждого из них, *вручную*. В случае данного workshop используется готовый файл, расположенный в папке *IHS\Configs* - переместите его в директорию Инъектора.

**Q11:** *Name for the target specific config for plugin "SpecFlow" (contains test assembly, etc). It is better to use "plug\_" prefix.*

**A11:** *plug\_IHS\_SpecFlow*

Далее программа запросит другой плагин, но мы заканчиваем их выбор, введя *ok*.

Следующий шаг: конфигурирование *дополнительного* логирования. Его можно пропустить, так как в любом случае логи будут писаться в локальную папку *logs\_drill* (на этапе инъекции - в директории *Инъектора*). Тем не менее можно указать *дублирующий* лог-файл, который будет писаться в общую для всех программ директорию. Здесь мы не будем этого делать.

**Q12:** *The Injector logs will be output to the its console and to a file in the its logs\_drill folder. Add an additional parallel log file? [n]*

**A12:** (клавиша Ввод)

Конфигурирование само по себе завершено и теперь будет запрос имени файла. Можно оставить имя по умолчанию - для *всех* подсистем *Drill4Net* оно *cfg.yml*. Однако можно ввести и произвольное - параллельно будет создан так называемый редирект-файл *\_redirect.yml*, в единственном свойстве *Path* которого описывается имя, либо произвольный путь к созданному конфигу (абсолютный или относительный - в данном случае для Инъектора). Если такой файл уже существует, возникнет запрос на его перезапись.

**Q13:** *Name of the Injector config: [cfg.yml]*

**A13:** *inj\_IHS*

Создание конфига завершено - Конфигуратор выдаст полный путь к файлу для проверки, если необходимо.



Здесь мы подготовили конфиг для инструментирования только *первого* билда. В рамках данного Workshop после этого нужно повторить все эти шаги для *второго* билда, указав его актуальную директорию (d:\Projects\IHS-bdd-2\), версию (0.2.0) и имя конфига (*inj\_IHS\_2*). Можно также скопировать первый конфиг под новым именем и затем изменить его параметры командой *"trg edit -l" или "trg edit --cfg\_path=inj\_IHS\_2"*. Или сделать это прямым редактированием нового конфига, что в данном случае намного проще.

Далее возникает два варианта развития событий, в зависимости от ваших потребностей. Как правило, отдельное инструментированное конечному пользователю не нужно, но может быть полезным для разработчика *Drill4Net*, тестировщику этой системы или QA-инженеру. В обычном воркфлоу следует выбирать *Подвариант A1 (CI)*.

## Подвариант А1

Если в планах создание полноценного *pipeline CI*, а обычно так и есть, если вы не разработчик самого Drill4Net, то на этом этапе больше ничего делать не нужно. Инструментированный таргет пока не существует, но он будет создан позже автоматически, после дальнейшего конфигурирования, описанного в соответствующих разделах.

## Подвариант А2

При необходимости мы можем провести *немедленное* инструментирование таргетов, запустив следующие команды:

```
trg inj --cfg_path=inj_IHS
trg inj --cfg_path=inj_IHS_2
```

А если инструментацию проводить сразу же после создания соответствующего конфига, то достаточно выполнять более простую команду для указания "последнего изменённого конфига" (см. хелп по командам):

```
trg inj -l
```

Обработка каждого таргета занимает около минуты, в результате чего рядом с исходными папками будут созданы две новые:

```
d:\Projects\IHS-bdd.Injecteд\
d:\Projects\IHS-bdd-2.Injecteд\
```

Теперь необходимо запустить в Конфигураторе команду для подготовки дальнейшего запуска таргета:

```
run prep -l (      )
run prep -- "d:\configs\injections\inj_IHS.yml" (    -      )
```

Данная команда копирует в директорию таргета специальный *агентский* конфиг, необходимый транзиттеру для подключения к админке *Drill*. Именно этот компонент передаёт сервису данные об отработавших частях кода ("пробы").



В случае формирования *CI pipeline* команда подготовки запуска выполняется автоматически.

## Вариант Б: Использование готовых конфигов

Для инструментации билдов *IHS* необходимо *последовательно* запустить консольную программу *Drill4Net.Injector.App.exe* из папки *apps/injector*, с разным указанием имени конфига в локальном файле *\_redirect.yml* (если он отсутствует, его нужно создать). В первом случае содержимое редирект-конфига должно быть "*Path: inj\_IHS\_Kafka*", а вторым "*Path: inj\_IHS\_Kafka\_2*" - это имена конфигов, которые *в данном случае* лежат здесь же, в папке Инъектора. В них самих необходимо установить *реальный* путь к соответствующему таргету, за который отвечает параметр *Source/Directory*.



В свойствах *Target/Version* указаны версии - 0.1.0 и 0.2.0 соответственно, а в *Target/Name* общее имя таргета - *IHS-bdd*.

Обработка каждого таргета занимает около минуты, в результате чего рядом с исходными папками будут созданы две новые:

```
d:\Projects\IHS-bdd.Injecteд\
d:\Projects\IHS-bdd-2.Injecteд\
```

В корне каждого билда должны появиться два файла: *injected.tree* (метаданные об инъекции) и *injected\_hint.tree* (текстовый файл с полным путём к нему).



Для продолжения работы необходимо разместить агентский файл конфигурации *agent.yml* из папки *install* в корень *каждой* из инструментированных папок, переименовав его в *cfg.yml*.

Этот агентский конфиг нужно настроить:



1. В свойстве *Admin/Url* указать реальный адрес подключения к админке *Drill* (согласно параметра *drill-admin/ports* компоуз-файла). По умолчанию - *localhost:8090*.
2. В свойстве *PluginDir* указать путь к *плагинам* транзиттера (реализующим интерфейс *IEngineContexter* - для работы с *Xunit*, *NUnit*, *MsTest* и т.д.).

На этот момент исходный таргет инструментирован и готов к запуску: прохождению тестов и сбору необходимой информации в админке.

## C. Workshop: tests

# 1. Configuring

Конфиг запуска можно создать через [Конфигуратор](#), либо создать конфиг вручную (в случае данного Workshop используются готовые).

## Вариант А: Конфигуратор

Необходимо запустить программу `d4n.exe` из папки `apps\configurator`, ввести команду `"new run"` (без кавычек) и ответить на несколько вопросов.

**Q1:** *Run's description:*

**A1:** Run the tests for IHS target (можно оставить пустым)

Далее будет задано несколько вопросов, связанных с возможностью параллельного выполнения тестов конкретного таргета (в случае Workshop это BDD-тесты на базе *SpecFlow* проекта *IHS Investor Access*). Иногда тесты можно выполнять только последовательно (либо из-за ресурсов, которые они потребляют, либо из-за ограничений тестовых фреймворков или самого Drill4Net).



Заметьте, что первые вопросы *запоминают* лишь значения *по умолчанию*, которые просто будут подставляться в вопросы как дефолт, в квадратных скобках - это призвано ускорить ввод *последующих* значений, и не более того. Затем, на конкретных папках и сборках, вы сможете указать конкретные значения, специфичные для каждой ситуации.

**Q2:** *Does it need to limit the parallel execution of tests in this run by DEFAULT? [n]*

**A2:** y

**Q3:** *Degree of parallelism (default): [8]*

**A3:** (число по умолчанию или чуть меньше, означающее количество параллельных тестов)

Теперь нужно ответить на вопрос, какое дефолтовое значение подставлять далее по такому ограничению? Если таргет в том или ином виде использует фреймворк xUnit, то ответ "y" (то есть, необходимо только последовательное исполнение). В любом случае, это всего лишь дальнейшая *дефолтовая подсказка*, которую в следующих вопросах, на конкретных сборках и тестах, можно изменить.

Далее следует указать директорию *инструментированных тестов*.

**Q4:** *Now you need to specify one or more tests' assemblies to run their tests. They can be located either in one folder or in several. To finish, just enter "ok".*

*Specify at least one tests' assembly.*

*Tests' directory:*

**A4:** d:\Projects\IHS-bdd.Injected\

Указываем уровень параллельности по умолчанию *на уровне папки* (так как в ней может находиться несколько файлов с тестами, различных по возможностям).

**Q5:** *Does it need to limit the parallel execution of tests in this FOLDER by DEFAULT? It contains Xunit 2.x tests? [y]*

**A5:** (клавиша Ввод)

**Q6:** *Tests' assembly name:*

**A6:** Ipreo.Csp.IaDeal.Api.Bdd.Tests.dll

Теперь указание запрета параллельности собственно тестов внутри указанной выше сборки.

**Q7:** *Does it need to limit the parallel execution of tests in this ASSEMBLY? It contains Xunit 2.x tests? [y]*

**A7:** (клавиша Ввод)

Теоритически можно добавить ещё одну сборку (dll) с тестами, находящуюся в указанной выше папке, но в рамках Workshop этого не нужно:

**Q8:** *One more tests' assembly name:*

**A8:** ok

Можно добавить ещё одну директорию с тестовыми сборками, но и этого в рамках Workshop не нужно (вводим *ok*):

**Q9:** *One more tests' directory:*

A9: ok

Q10: Name of the TestRunner config: [cfg.yml]

A10: run\_IHS

Правила именования и сохранения те же, что и в случае создания конфига инъекции.



Здесь мы подготовили конфиг для запуска *первого* билда. В рамках данного Workshop после этого нужно повторить все эти шаги для *второго* билда, указав его актуальную *инструментированную* директорию (d:\Projects\IHS-bdd-2.Injected\). Можно также скопировать первый созданный конфиг под новым именем и затем изменить его параметры командой `"run edit -l"` или `"run edit --cfg_path=run_IHS_2"`. Или сделать это прямым редактированием нового конфига.

## Вариант Б: Готовые конфиги для Workshop

Для первого билда это должен быть `run_IHS.yml`, указанный в `_redirect.yml`. Оба файла должны лежать в директории `Test Runner` (по умолчанию - `apps/test_runner`).

## 2. Running the tests

Для запуска автоматических тестов таргета используется программа *Test Runner*, сконфигурированная для *конкретного* таргета. Программа автоматически определит, какие тесты нужно запустить, где они расположены и прочие параметры. Если для админки Drill текущий таргет является новым, будут запущены *все* тесты. Если тестов, *необходимых* для запуска нет, программа закроется (можно посмотреть логи). В противном случае запустятся лишь те, которые покрывают изменённые и новые методы текущего билда.



Выяснив информацию о тестах, *Test Runner* запускает стандартную консоль *VSTest* (установлена в SDK), после чего он закрывается.

Запуск тестов можно произвести *непосредственно*, запустив *Test Runner* вручную или через Конфигуратор, а можно [сконфигурировать](#) и далее использовать *pipeline CI*. В первых случаях нужно проследить нахождение агентского конфига в папке каждого задействованного таргета - в Конфигураторе для этого можно использовать команду *run prep* (см. внутренний мануал команды).



Настоятельно рекомендуется использовать полный [воркфлоу CI](#).

### Вариант А: Test Runner (Конфигуратор)

```
run test -l (      Test Runner)
run test -a (      - Test Runner)
run test -- "d:\configs\test_runner\run_cfg.yml" (      )
run test --cfg_path="d:\configs\test_runner\run_cfg.yml" (      )
```

### Вариант Б: Test Runner (напрямую)

Для IHS-практикума *Test Runner* можно запускать просто по клику мышкой на его иконке, без аргументов командной строки - конфиг "подхватится" согласно редирект-файла. Для первого билда это должен быть *run\_IHS.yml*, указанный в *\_redirect.yml*. Оба файла должны лежать в директории *Test Runner* (по умолчанию - *apps/test\_runner*).

Следует как минимум проверить путь к инструментированной сборке в *актуальном* конфиге *Test Runner* - свойство *Directory*. Для первого запуска следует проконтролировать параметры тестовой сборки (*DefaultAssemblyName* и *DefaultParallelRestrict*), а также адреса сервера Кафки в конфиге Трансммиттера (свойство *Servers*). Для практикума проекта *IHS* эти значения выставлены по умолчанию.

### Вариант В: CI (Конфигуратор)

Если соответствующие конфиги CI были [сконфигурированы](#), то полный цикл обработки, включая инструментирование исходных таргетов, можно запустить одной из перечисленных команд:

```
ci start -l (      CI)
ci start -a (      - CI)
ci start -- "d:\configs\ci\ci_1.yml" (      )
ci start --cfg_path="d:\configs\ci\ci_1.yml" (      )
```

## 3. CI (optional)

Если было принято решение использовать полный *pipeline CI*, то его также необходимо сконфигурировать - с использованием уже подготовленных конфигов для инструментирования и последующего запуска тестов.

Общий смысл конфигурирования CI такой:

1. Сам по себе процесс включает в себя два этапа обработки: инструментирование указанных таргетов (инъекции) и последующий запуск находящихся в них автоматических тестов тем или иным способом.
2. Удобно создать общую папку для различных *pipelines* (по сути, таргетов). По умолчанию она находится в папке дистрибутива и называется *ci*. В ней стоит создать отдельные папки конкретных *pipelines*. Дальнейшая организация внутренних папок и файлов довольно гибка. В корне CI можно использовать редирект-конфиги и остальные конфиги, имеющие *тип CI*, могут лежать тут же, а можно хранить отдельные CI-конфиги в папках конкретных pipeline.
3. В одном *CI pipeline* можно проводить инъекции для нескольких совершенно независимых таргетов - по соответствующим конфигам (то есть, их *несколько*). Их можно выложить в папку конкретного pipeline, находящуюся в свою очередь в корневой папке CI (\apps\ci по умолчанию), но допускается и в любое другое место, однако сущности одного процесса всё же лучше держать вместе.
4. Одним *CI pipeline* можно запускать тесты из различных, несвязанных между собой инструментированных сборок и папок, но для этого используется *один-единственный* конфиг типа *TestRunner* (потому что он *позволяет* это сделать и к тому же является относительно простым - в отличие от конфига инъекции).
5. Собственно *CI pipeline* можно запускать из Конфигуратора напрямую, а также интегрировать этот запуск в сторонние системы. На данный момент это интеграция в среды разработки (Visual Studio, JetBrains Rider и т.д.), точнее, в файлы .NET-проектов. Это позволяет автоматически запускать все необходимые процессы сразу же после компиляции. В будущем планируется реализовать подходы для TeamCity, Jenkins и т.д.

Предположим, что общая папка CI общая и стандартная - D:\Drill4Net\apps\ci. Делаем в ней папку *данного* pipeline: *IHS\_workshop*, а внутри неё - папку *injections* для конфигов Инъектора. В папку *injections* разместите имеющиеся конфиги инъекций двух билдов (*inj\_IHS.yml* и *inj\_IHS\_2.yml*), а в корень папки *IHS\_workshop* - сконфигурированный ранее конфиг для *Test Runner*.



Для настройки путей к подпрограммам Drill4Net и рабочим папкам можно отредактировать конфиг конфигулятора *app.yml*. Восстановить значения по умолчанию можно командой "cfg restore".

Далее необходимо запустить [Конфигуратор](#) - *d4n.exe* из папки *apps\configurator*, ввести команду *"new ci -I"* (без кавычек) и ответить на несколько вопросов.

**Q1:** *CI run's description: [Example configuration for the end2end workflow - it runs batch injections of some targets and then tests using TestRunner]*

**A1:** *CI pipeline for the IHS tests (можно оставить пустым)*

**Q2:** *The Run have to has the one or more Injector configs. Specify the directory with them (they will all be used):*

**A2:** *D:\Drill4Net\apps\ci\IHS\_workshop\injections*

**Q3:** *The degree of parallelism on level those configs: [8]*

**A3:** *(оставить по умолчанию или сделать чуть меньше)*

**Q4:** *TestRunner config path to run the injected targets:*

**A4:** *D:\Drill4Net\apps\ci\IHS\_workshop\run\_IHS.yml*

**Q5:** *Config path for this CI run will be saved to: [D:\Drill4Net\apps\ci\IHS\_workshop\ci.yml]*

**A5:** *(тут много возможностей: как минимум согласится на дефолт или сохранить на уровень выше, но там нужно будет использовать и учитывать редирект-конфиг *\_redirect.yml*)*

После этого конфиг CI будет сохранён. Если при вызове команды не использовать переключатель *-I*, вас спросят о необходимости *интеграции* CI pipeline - для данного workshop этот шаг необходимо пропустить, и нужно будет отказаться.

Теперь всё готово, и можно запускать всю цепочку CI.

# 4. Drill Web App

## Регистрация нового таргета

Страница веб-приложения админки Drill сразу после установки по дефолтовому адресу <http://localhost:8091/>:

agents

0

Signed in as Guest [Sign out](#)

Agents 0

Preregister Offline Agent

Подключение к системе может занять до 15 секунд.

Если таргет новый, окно *Test Runner* будет "вечно висеть". Нужно зайти в админку и зарегистрировать агента.

Запущенный вручную или с помощью [Конфигуратор](#) окно *Test Runner*:

```
D:\Projects\EPM-D4J\Drill4Net\build\bin\Debug\Drill4Net.Agent.TestRunner\net5.0\Drill4Net.Agent.TestRunner.exe
2021-11-18 10:08:04.120 Information|Main|Starting...
2021-11-18 10:08:04.474 Debug|BaseOptionsHelper|1|ReadOptions|Reading config: [D:\Projects\EPM-D4J\Drill4Net\build\bin\Debug\Drill4Net.Agent.TestRunner\net5.0\IHS.yml]
2021-11-18 10:08:04.487 Debug|BaseOptionsHelper|1|ReadOptions|Config deserialized.
2021-11-18 10:08:04.488 Debug|BaseOptionsHelper|1|ReadOptions|Config prepared.
2021-11-18 10:08:04.492 Debug|AgentTestRunner|TreeRepositoryHelper|CalculateTreeFilePath|The tree hint file founded: [d:\Projects\IHS-bdd.Injected\Injected_hint.tree]
2021-11-18 10:08:04.501 Debug|BaseOptionsHelper|1|ReadOptions|Reading config: [D:\Projects\EPM-D4J\Drill4Net\build\bin\Debug\Drill4Net.Agent.TestRunner\net5.0\cfg.yml]
2021-11-18 10:08:04.502 Debug|BaseOptionsHelper|1|ReadOptions|Config deserialized.
2021-11-18 10:08:04.502 Debug|BaseOptionsHelper|1|ReadOptions|Config prepared.
2021-11-18 10:08:04.504 Debug|Agent|TreeRepository|2|CheckTreeFilePath|Path param = [d:\Projects\IHS-bdd.Injected\Injected.tree]
2021-11-18 10:08:04.508 Debug|Agent|TreeRepository|2|ReadInjectedTree|The tree file will be read: [d:\Projects\IHS-bdd.Injected\Injected.tree]
2021-11-18 10:08:05.158 Debug|Agent|StandardAgentRepository|Init|Creating...
2021-11-18 10:08:05.166 Information|Agent|ContextDispatcher|.ctor|Plugin added (standard): [SimpleContexter]
2021-11-18 10:08:05.178 Information|Agent|StandardAgentRepository|Init|Target: [IHS-bdd] version: 0.1.0
2021-11-18 10:08:06.569 Information|Agent|AdminRequester|GetData|New target for Drill: IHS-bdd
2021-11-18 10:08:06.623 Debug|Agent|StandardAgentRepository|GetConnectorLogParameters|Connector's logging: [Debug] to [D:\Projects\EPM-D4J\Drill4Net\build\bin\Debug\Drill4Net.Agent.TestRunner\net5.0\connector.log]
2021-11-18 10:08:06.693 Debug|Agent|StandardAgentRepository|Init|Created.
2021-11-18 10:08:06.883 Debug|Agent|StandardAgent|["PID":38216].ctor|StandardAgent is initializing...
2021-11-18 10:08:06.885 Debug|Agent|StandardAgent|["PID":38216].ctor|Initializing is fast: False
2021-11-18 10:08:06.889 Information|Agent|Communicator|Connect|Connect
2021-11-18 07:08:07 [WINDOWS][INFO][NativeAgentLibrary] Admin address: localhost:8090
2021-11-18 07:08:07 [WINDOWS][INFO][NativeAgentLibrary] Agent config: AgentConfig(id=IHS-bdd, instanceId=23e69450-354e-4f69-82a8-35441793b70c, buildVersion=0.1.0, serviceNeedSync=true, packagesPrefixes=PackagesPrefixes(packagesPrefixes=[]), parameters=(logLevel=AgentParameter(type=string, value=DEBUG, description=Logging agent work. can't be changed, value=D:\Projects\EPM-D4J\Drill4Net\build\bin\Debug\Drill4Net.Agent.TestRunner\net5.0\connector.log, description=the location where the logs will be stored)))
Configuration for mingw
2021-11-18 10:08:07.513 Debug|Agent|StandardAgent|["PID":38216].ctor|StandardAgent is primarily initialized.
2021-11-18 10:08:07.533 Debug|AgentTestRunner|Runner|Run|Wait for Agent's initializing...
2021-11-18 07:08:07 [WINDOWS][DEBUG][Transport] successfully connected
2021-11-18 07:08:07 [WINDOWS][INFO][DrillWebsocket] Agent connected with instanceId '23e69450-354e-4f69-82a8-35441793b70c'
2021-11-18 07:08:08 [WINDOWS][INFO][tempTopicLogger] Agent got a new headerMapping:
2021-11-18 07:08:08 [WINDOWS][DEBUG][DrillWebsocket] '/agent/change-header-name' took 944us
2021-11-18 10:08:08.138 Debug|Agent|AgentReceiver|MessageReceived|Message received: topic=/agent/change-header-name, message=
2021-11-18 07:08:08 [WINDOWS][DEBUG][Transport] sent 40
```

После того, как *Test Runner* подключится, в админке появится незарегистрированный агент. Его идентификатор - это имя *таргета* из конфига. В случае, когда последний админке ещё не известен, *Test Runner* заблокирует своё выполнение до момента ручной регистрации агента (таргета) в веб-приложении админки.

Существует возможность предварительной регистрации агента.

agents

0

Signed in as Guest [Sign out](#)

Agents 1

Preregister Offline Agent

Name	Description	Type	Environment	Status	
<div><div></div><div>New</div><div>IHS-bdd</div></div>		.NET	n/a	<div><div></div>Off</div>	<div>Register</div>

Следует нажать кнопку *Register*. В первом окне можно ничего не менять (при необходимости лишь изменить описание).



Версия агента здесь - это версия *компонента Agent*, а не версия таргета как такового.

agents / IHS-bdd / registration



Signed in as Guest [Sign out](#)

## .NET Agent Registration

[Abort Registration](#)

### 1 of 3. General Settings

[Continue >](#)

① Set up basic agent settings.

Agent ID

IHS-bdd

Agent version

0.8.108-main%2bacff6b7

Service Group

n/a

Agent name

IHS-bdd

Description

Optional

Add agent's description

Environment

Optional

Specify an environment

Нажимаем *Continue*, затем ещё раз (вводимые данные Drill4Net пока не используют).

agents / IHS-bdd / registration



Signed in as Guest [Sign out](#)

## .NET Agent Registration

[Abort Registration](#)

### 3 of 3. Plugins

[< Back](#)

[✓ Finish](#)

① Choose plugins to install on your agent. You will also be able to add them later on Agent's page.

1 of 1 selected



Test2Code (0.8.0-31)


Test2Code plugin minimizes your regression suite via Test Impact Analytics by suggesting only affected subset of tests to run, and highlight untested areas via Test Gap Analysis, providing evidence of how changes are tested and which areas and not tested at all.


Здесь должен быть виден плагин *расчёта coverage* с названием *Test2Code*, находящийся на стороне админки - галка должна быть выбрана. Заканчиваем регистрацию.


Если же регистрация уже была проведена, то при подключении либо таргета, либо *Test Runner* мы попадаем в *Dashboard*, но для самого первого запуска данных ещё нет - цифры здесь появятся только после *полного* прохождения тестов.





[agents](#) / [IHS-bdd](#) / [builds](#) / [0.1.0](#) / [dashboard](#)

 0 | Signed in as Guest [Sign out](#)

 **IHS-bdd**

Online

 Dashboard



TEST2CODE

[View more >](#)

Build Coverage

0%

Tests

0



0 scopes

Auto

Manual

Risks

0

Tests to run


0


Auto

Manual


По клику на нижней, крайней иконке слева попадаем на вкладку плагина *Test2Code*:


[agents](#) / [IHS-bdd](#) / [builds](#) / [0.1.0](#) / [dashboard](#) / [test2code](#) / [overview](#)


 0 | Signed in as Guest [Sign out](#)

 **IHS-bdd**


Online



 Test2Code

Current build: ... 

Parent build: -

QUALITY GATE 



[Configure](#)

RISKS

—

TESTS TO RUN

—

 **fx Build methods**  Build tests

BUILD COVERAGE

● Build

● Build / Active Scope overlap (0%)

● Active Scope unique coverage (+0%)

0% Press "Finish Scope" button to add your scope coverage to the build.

0 25 50 75 100

TOTAL METHODS 2211

NEW 0


MODIFIED 0

0 0 deleted

0

0

ACTIVE SCOPE COVERAGE


0% 






[Finish Scope](#)


Scope Details

All Scopes

Sessions Management









 Search packages by name

Name	Coverage, % 	Methods total	Methods covered	Associated tests
<a href="#">&gt;</a>  Ipreo/NS/InvestorAccess/ServiceCall/Models	0%	22	0	n/a
<a href="#">&gt;</a>  Ipreo/NS/BuySide/BuildingBlocks/Framework/ExceptionHandling/Exceptions	0%	26	0	n/a
<a href="#">&gt;</a>  Ipreo/NS/BuySide/BuildingBlocks/Framework/Http/Identity	0%	14	0	n/a
<a href="#">&gt;</a>  Ipreo/Csp/IaDeal/Api/Controllers	0%	64	0	n/a

 Админке потребуется несколько секунд, чтобы получить от агента данные о классах таргета и вывести их таблицу (в это время статус веб-приложения будет "Busy").


Следует обратить внимание на ключевые области страницы: Статус, вкладки "Build methods" и "Build tests" слева, ссылки "Risks" и "Tests to run" (справа) и блок "Active scope coverage" (чуть ниже). В последнем можно заметить ссылку "Scope details" - именно там будут отображаться данные по текущему запуску в реальном времени: покрытие тестами кода и собственно найденные тесты.

Методы группируются по *namespace* и *классам*:


Name
>  Ipreo/NS/InvestorAccess/ServiceCall/Models
>  Ipreo/NS/BuySide/BuildingBlocks/Framework/ExceptionHandling/Exceptions
>  Ipreo/NS/BuySide/BuildingBlocks/Framework/Http/Identity
▼  Ipreo/Csp/IaDeal/Api/Controllers
>  ActivityStreamController
▼  ComplianceAuditController
<div><div>fx</div><div>.ctor</div><div>(Ipreo.Csp.IaDeal.BusinessService.ComplianceAudit.GetAudit.GetDealAuditEventsOrchestrator, Ipreo.Csp.IaDeal.BusinessService.ComplianceAudit.GetAud...</div></div>
<div><div>fx</div><div>GetAuditIdsAsync</div><div>(Ipreo.Csp.IaDeal.Model.ComplianceAudit.SearchAuditIdDto):System.Threading.Tasks.Task`1&lt;Microsoft.AspNetCore.Mvc.IActionResult&gt;</div></div>
<div><div>fx</div><div>GetDealAuditEventsAsync</div><div>(Ipreo.Csp.IaDeal.Model.ComplianceAudit.DealAuditEventQueryParamsDto):System.Threading.Tasks.Task`1&lt;Microsoft.AspNetCore.Mvc.IActionResult&gt;</div></div>
<div><div>fx</div><div>GetXsdDealAuditEventsAsync</div><div>():Microsoft.AspNetCore.Mvc.IActionResult</div></div>
<div><div>fx</div><div>ResubmitAuditEventsAsync</div><div>(System.Collections.Generic.IReadOnlyCollection`1&lt;System.Int32&gt;):System.Threading.Tasks.Task`1&lt;Microsoft.AspNetCore.Mvc.IActionResult&gt;</div></div>
>  DealAuditController
>  DealController




На описанную страницу можно зайти и со страницы агентов - при щелчке на имени.

agents

 0 | Signed in as Guest [Sign out](#)

Agents 1

 Preregister Offline Agent

Name	Description	Type	Environment	Status	
 IHS-bdd		.NET	n/a	 On	

## Обработка текущего билда

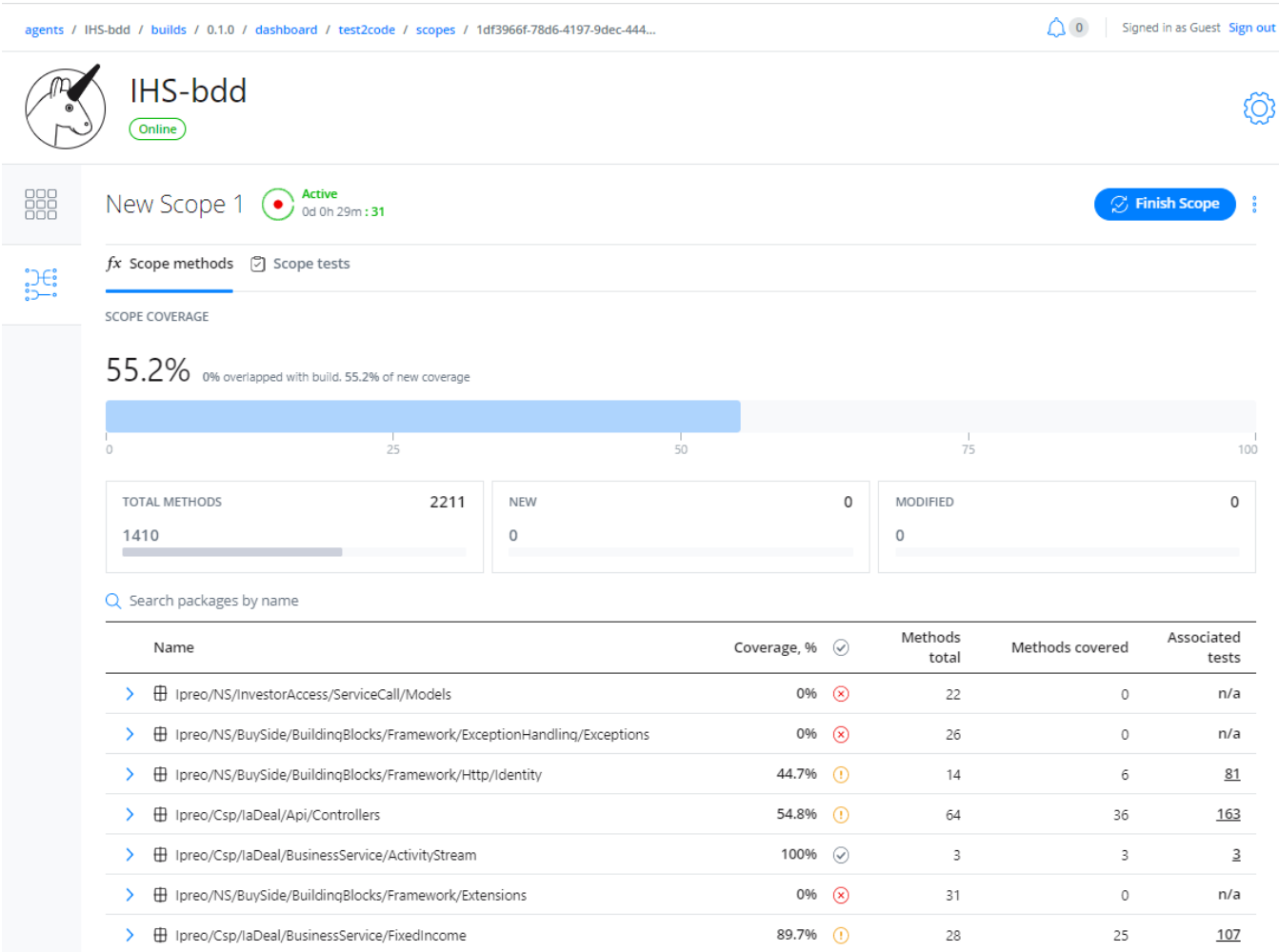
После обработки админкой данных о классах инициализация считается завершенной, и заблокированный *Test Runner* запускает стандартную консоль *Microsoft VS Test* - она и будет выполнять тесты из тестовой сборки проекта. *Test Runner* при этом закроется, и теперь к админке подключится выполняющийся *target*. Через некоторое время автоматически начнется *сессия* - это видно по иконке с красной точкой справа. Это означает, что админка готова к приёму проб от инструментированного приложения.

На картинке ниже можно видеть одновременно три окна: на заднем фоне админка *Drill*, вверху - *Agent Server* в консоли, а внизу - окно *VS Test*. Видно, что пробы поступают в текущий *score* (цифра справа), но слева, при наличии голубой полосы прогресса, наблюдается 0%. Это просто означает, что мы *не находимся в окне скоупа*. Войти в него можно по ссылке "Scope details" под кнопкой "Finish scope".

✔ Для наблюдения за прохождением тестов в реальном времени нужно перейти по ссылке "Scope details" справа.



Для наблюдения за прохождением тестов в реальном времени нужно перейти по ссылке "Scope details" справа.




На странице активного скоупа, на вкладке "Scope methods", можно видеть как постепенно растёт покрытие в процентах, а в списке методов увеличиваются цифры покрытия конкретных *методов*.

С другой стороны, на вкладке "Scope tests" появляются новые *тесты* с некоторыми данными об их прохождении.

agents / IHS-bdd / builds / 0.1.0 / dashboard / test2code / scopes / 1df3966f-78d6-4197-9dec-444...


0

Signed in as Guest [Sign out](#)



IHS-bdd

Online



New Scope 1

Active

0d 0h 27m : 25

Finish Scope

fx

Scope methods

☒ Scope tests

TESTS EXECUTION

Auto

Manual

289

tests executed in scope

AUTO

289

54.5% methods covered

MANUAL

0

0% methods covered

Search tests by name


Name	Test type	Status	Coverage, %	Methods covered	Duration
<input checked="" type="checkbox"/> Deal version without messages -	Auto	Passed	17.7	<u>302</u>	00:00:21
<input checked="" type="checkbox"/> Deal version with messages without permitted investor -	Auto	Passed	4.6	<u>184</u>	00:00:04
<input checked="" type="checkbox"/> Deal version without tranche -	Auto	Passed	5.7	<u>243</u>	00:00:05
<input checked="" type="checkbox"/> Get Chinese deal by deal id - return RegS - true, return latest - false -	Auto	Passed	2.8	<u>141</u>	00:00:06
<input checked="" type="checkbox"/> Get deal by deal id - return RegS - true, return latest - true -	Auto	Passed	2.8	<u>140</u>	00:00:05

После того, как необходимые тесты отработают (сейчас это около получаса для первого билда *IHS*), консоль *VS Test* закроется.

agents / IHS-bdd / builds / 0.1.0 / dashboard / test2code / scopes / 1df3966f-78d6-4197-9dec-444...


0

Signed in as Guest [Sign out](#)



IHS-bdd

Online




New Scope 1

Finished

0d 0h 35m

После окончания тестов (физического исчезновения *таргета*) коннект к админке ещё некоторое время будет открыт, и только потом *агент* будет считаться отключенным. Так происходит, потому что в серверном варианте инфраструктуры *Drill4Net* компонент *агента* находится не в таргете, а в отдельном процессе *Agent Worker*.




Админка не различает **таргет** как исследуемый проект и **агент** как его *профайлер*. Если для каких-то целей модуль агента подключается к админке от имени некоего таргета из *разных* программ, эти программы и будут для неё этим "таргетом", что технически не совсем верно.


*Agent Worker* автоматически закрывается сервером *Agent Server*, что можно увидеть в его логах:

```
2021-11-23 11:21:18 [WINDOWS][DEBUG][DrillWebsocket] '/plugin/action' took 2.47ms
2021-11-23 14:21:18.908[Debug]Agent|AbstractCoveragerSender|SendScopeInitialized|Send ScopeInitialized
2021-11-23 11:21:18 [WINDOWS][DEBUG][Transport] sent 219
2021-11-23 11:21:18 [WINDOWS][DEBUG][Transport] sent 29
2021-11-23 11:21:18 [WINDOWS][DEBUG][Transport] sent 212
2021-11-23 14:21:43.115[Information]AgentServer|AgentServer|CheckWorker|Closing worker: c03e760c-4bad-4b2c-8b35-46b0b0b5b016 -> IHS-bdd 0.1.0
```

Конечное состояние:

agents / IHS-bdd / builds / 0.1.0 / dashboard / test2code / scopes / 1df3966f-78d6-4197-9dec-444...


 IHS-bdd  
Offline





## Анализ результатов


Теперь *Dashboard* выглядит так:

agents / IHS-bdd / builds / 0.1.0 / dashboard

 IHS-bdd  
Offline



 Dashboard



TEST2CODE

Build Coverage

56.3%

Tests

364

1 scopes

Risks

0

Tests to run

0

Auto

Manual

Auto

Manual

View more >

Это означает, что покрытие тестами в данном скоупе (в текущем запуске) - 56,3% при отработке 364 тестов. Цифры рисков и рекомендуемых тестов пока по нулям, потому что отработал лишь первый билд, *базовый*.



Активного скоупа уже нет - он автоматически закрыт, но его данные перенесены на главную страницу билда (здесь его версия "0.1.0").

Теперь можно посмотреть, какие методы были затронуты определёнными тестами. Значения в колонке "Associated tests" кликабельны.



IHS-bdd

Offline



Test2Code

Current build: ...  
Parent build: -

RISKS

-

TESTS TO RUN

-

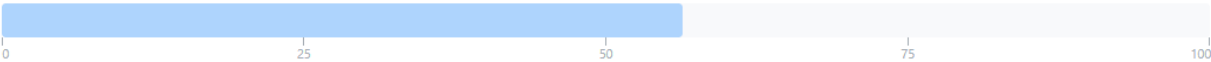


fx Build methods ☒ Build tests

BUILD COVERAGE

[All scopes](#)

56.3%



TOTAL METHODS	2211	NEW	0	MODIFIED	0
1438	0 deleted	0		0	

Search packages by name

Name	Coverage, %	Methods total	Methods covered	Associated tests
>  Ipreo/NS/InvestorAccess/ServiceCall/Models	0%	22	0	n/a
>  Ipreo/NS/BuySide/BuildingBlocks/Framework/ExceptionHandling/Exceptions	0%	26	0	n/a
>  Ipreo/NS/BuySide/BuildingBlocks/Framework/Http/Identity	44.7%	14	6	<u>87</u>
>  Ipreo/Csp/IaDeal/Api/Controllers	60.9%	64	41	<u>181</u>
>  Ipreo/Csp/IaDeal/BusinessService/ActivityStream	100%	3	3	<u>3</u>

×

Associated tests 87

Package

lpreo/NS/BuySide/BuildingBlocks/Frameworko...

All tests ▾

Q

Search tests by name

☒

Assign same IN id to 2 deals

-

☒

Bank disappears from Order managers when re...

-

☒

Billing Report - missing publisher scenario

-

☒

Billing Report - verify BookState dates

-

☒

Billing Report - verify IEnabled data

-

☒

Billing Report - verify dates

-

☒

Billing Report - verify deal data

-

☒

Billing Report - verify issuer data

-

☒

Billing Report - verify number of buyside comp...

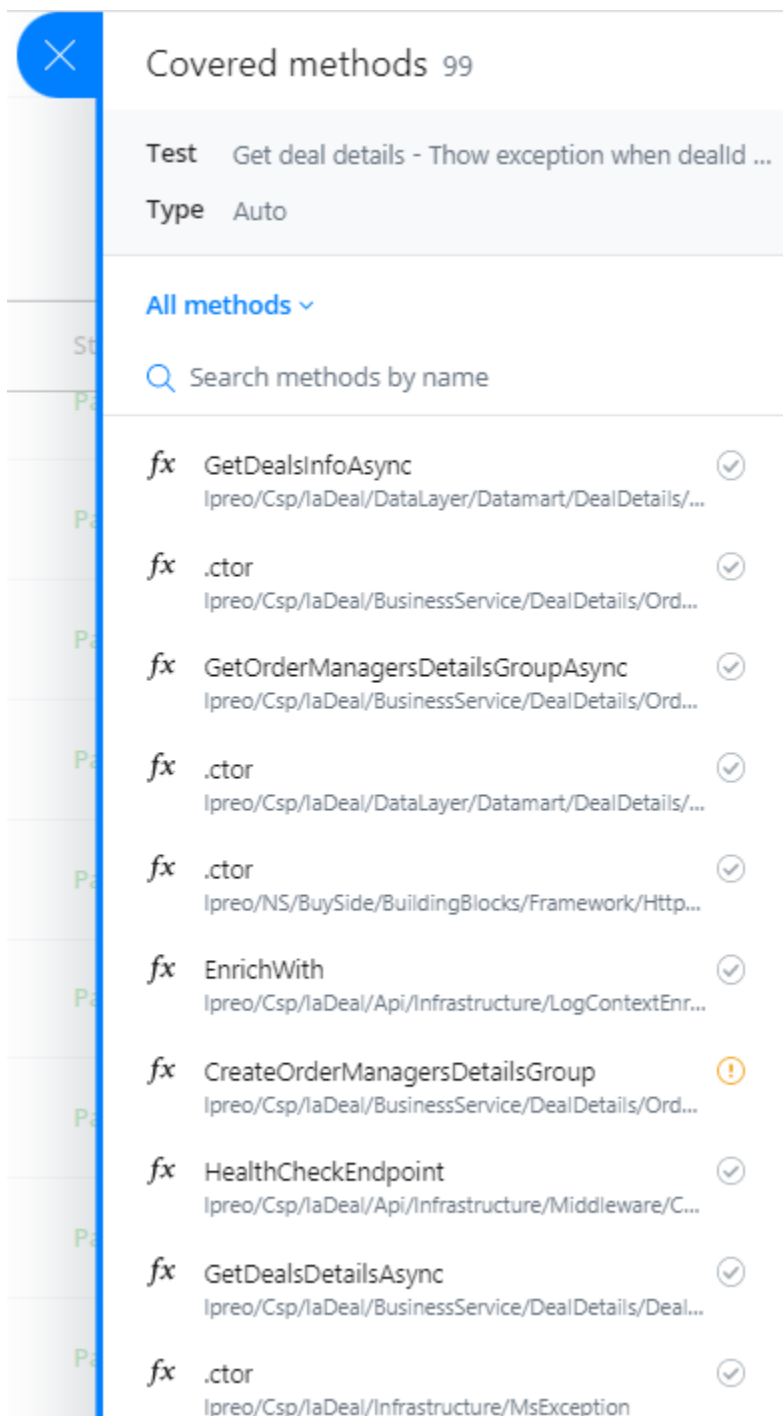
-



Пока тесты не отработали (то есть, сессия активна), данные по таким связям отображаться не будут.

Перейдя на страницу тестов можно изучить обратную зависимость: какие методы вызывал каждый тест.





## Второй билд таргета

### Первичный анализ


Когда программист получает новый билд искомого проекта, с новыми или изменёнными методами, следует повторить всё заново:

- провести инструментирование - скажем, с версией таргета "0.2.0", используя новый конфиг. Для воркшопа это *inj\_IHS\_Kafka\_2* - его имя следует прописать в *\_redirect.yml* папки *injector*.
- настроить конфиг *Test Runner* - в файле *\_redirect.yml* папки *test\_runner* прописать конфиг *run\_IHS\_2*. В нём самом указать папку второго инъецированного билда (если она сменилась) в свойстве *Directory*.


- провести запуск тестов с помощью *Test Runner*. Регистрации агента теперь не будет, а блокироваться будет сам *таргет* (процесс *VSTest*) - но лишь на некоторое время инициализации админки.

 В процессах CI/CD эти действия несложно [автоматизировать](#).

После подключения агента можно будет увидеть оповещение о том, что получена новая версия:

**1**


Signed in as Guest [Sign out](#)



RISKS

TESTS TO RUN

×

 Notifications

Mark all as read

Clear all


IHS-bdd


52 seconds ago

● Build 0.2.0 arrived

[Dashboard](#) [What's new](#)


Админка снова получит данные по его классам и вычислит, какие появились риски (новые/изменённые *методы*), а также какие тесты необходимо для них запустить - автоматически, с помощью *Test Runner*.


agents / IHS-bdd / builds / 0.2.0 / dashboard  0 Signed in as Guest [Sign out](#)




IHS-bdd

Online





Dashboard



TEST2CODE [View more >](#)

Build Coverage

0%

Tests

0



0 scopes

Auto

Manual

Risks

1



Tests to run

3

Auto

Manual

Данные по рискам и рекомендуемым тестам появятся и на странице плагина *Test2Code* (вверху справа):



IHS-bdd

Online



## Test2Code

Current build: ...  
Parent build: ...QUALITY GATE ⓘ  
[Configure](#)RISKS  
1 >TESTS TO RUN  
3 >fx Build methods ☒ Build tests

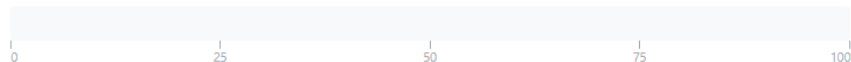
Test2Code

## BUILD COVERAGE

● Build ● Build / Active Scope overlap (0%) ● Active Scope unique coverage (+0%)

0%

Press "Finish Scope" button to add your scope coverage to the build.



TOTAL METHODS

2211

NEW

0

MODIFIED

1

0

0 deleted

0

0

1 risks

## ACTIVE SCOPE COVERAGE

0%

[Finish Scope](#)[Scope Details](#)[All Scopes](#)[Sessions Management](#)

Пройдя по ссылке "Risks", можно увидеть конкретные методы, для которых нужно запустить некоторые тесты (физически имеющиеся в таргете, а не "вообще в теории"). Так, во втором билде был изменён лишь один метод - *ContainsProperties* в классе *ILMapperExtensions*.



IHS-bdd

Online



## Risks 1

Build: 0.2.0 Compared to: Build 0.1.0



ALL RISK METHODS (1)

[Search methods by name](#)

Name	Type	Coverage, %	Associated Tests
fx ContainsProperties Ipreo/Csp/IaDeal/BusinessService/IL/Mappings/ILMapperExtensions	Modified	0%	0

Пройдя по ссылке "Tests to run", можно увидеть рекомендуемые тесты и время, которое будет сэкономлено против полного запуска всех тестов. В данном случае необходимо выполнить всего лишь три теста.



IHS-bdd

Online



## Tests to Run 3

Build: 0.2.0 Compared to: ...

[Get Suggested Tests](#)

TOTAL DURATION

00:33:00

ESTIMATED TIME SAVED

00:32:49 | 99.4%

TOTAL TIME SAVED

--:--:--



## SAVED TIME HISTORY

No data Saved time Duration with Drill4J

40m

30m

20m

10m

0

Build 0.2.0

## ALL SUGGESTED TESTS (3)

[Search tests by name](#)

Name	Test type	State	Coverage, %	Methods covered	Duration
<input checked="" type="checkbox"/> Should add consumed deal publish record to DB during publish message -	Auto	To run			
<input checked="" type="checkbox"/> Get current published message in the audit -	Auto	To run			
<input checked="" type="checkbox"/> IL deal workflow - Distr type some -	Auto	To run			

Test Runner согласно этим данным запустит VS Test, а сам закроется.

## Прохождение тестов второго билда

Тесты второго билда начинаются также в VS Test, автоматически. Активный скоуп будет содержать только рекомендованные тесты - в данном случае, лишь три теста, после чего таргет и агент отключатся, а новые данные сольются на странице плагина Test2Code со старыми.

fx Scope methods ☒ Scope tests

## TESTS EXECUTION

Auto Manual

3 tests executed in scope




AUTO	3	MANUAL	0
37% methods covered		0% methods covered	

[Search tests by name](#)

Name	Test type	Status	Coverage, %	Methods covered	Duration
<input checked="" type="checkbox"/> Get current published message in the audit -	Auto	Passed	22.3	737	00:00:13
<input checked="" type="checkbox"/> IL deal workflow - Distr type some -	Auto	Passed	20.6	814	00:00:03
<input checked="" type="checkbox"/> Should add consumed deal publish record to DB during publish message -	Auto	Passed	32.1	869	00:00:04


## Анализ прохождения тестов второго билда


Если всё прошло удачно, риски исчезнут (к этому и нужно стремиться):




IHS-bdd

Offline



Dashboard



TEST2CODE

Build Coverage

37%

- 19.3% vs Build 0.1.0

Tests

3



1 scopes

Auto

Manual

Risks

0



Tests to run

0


Auto

Manual

[View more >](#)


На данный момент считается, что статусы тестов (прохождение или провал) сами по себе не влияют на снятие риска, важен лишь факт отработки теста. Если тест технически отработал - риск неизвестности снят.


На странице "Tests to run" видны времена и график сэкономленного времени.



IHS-bdd

Offline





Tests to Run 0

Build: 0.2.0 Compared to: ...

[Get Suggested Tests](#)

TOTAL DURATION


00:33:00

ESTIMATED TIME SAVED


00:32:49 | 99.4%

TOTAL TIME SAVED

00:32:41 | 99%



SAVED TIME HISTORY



ALL SUGGESTED TESTS (3)

Name	Test type	State	Coverage, %	Methods covered	Duration
<input checked="" type="checkbox"/> Should add consumed deal publish record to DB during publish message	Auto	Done	32.1	869	00:00:04
<input checked="" type="checkbox"/> Get current published message in the audit	Auto	Done	22.3	737	00:00:13
<input checked="" type="checkbox"/> IL deal workflow - Distr type some	Auto	Done	20.6	814	00:00:03

Сейчас данные по "тайм-сэвингам" считаются относительно чистого времени прохождения собственно тестов, то есть не учитываются времена: инъекции исходных сборок, регистрации агента, инициализации админки, подготовки тестовых фреймворков, тестовых классов и тестов по отдельности.

Примечания

- Теперь можно будет посещать страницы как первого билда, так и всех последующих:



[builds](#)  
**IHS-bdd**  
Offline

## All builds 2


Name	Added
<a href="#">0.1.0</a>	Nov 18, 2021 at 9:09 AM
<a href="#">0.2.0</a>	Nov 18, 2021 at 9:52 AM

- При щелчке на странице агентов по имени таргета мы будем переходить в самый последний билд.



## Agents 1

[Preregister Offline Agent](#)

Name	Description	Type	Environment	Status
 <a href="#">IHS-bdd</a>  <a href="#">IHS-bdd</a>		.NET	n/a	<input type="checkbox"/> Off

- После изучения результатов текущего билда стоит делать базовым (*baseline*), чтобы изменения и рекомендации для следующего билда рассчитывались относительно него, а не "самого первого".




**IHS-bdd**  
Offline

Set as Baseline



Test2Code

Current build: ... 

Parent build: ... [0.2.0](#)



**Build methods** ☒ **Build tests**

BUILD COVERAGE