Теперь вы знаете о своем бизнесе ВСЕ

iikoCard™

API: интерфейс программирования приложений

iikoCard.API

Содержание

Глава 1

iikoCard.API

Введение

- Далее дано описание клиентского API к серверу iikoCard.
- Клиентский API реализован в модуле **Resto.Card.Client.Library.dll**.
- Сборка устанавливается из *<Distrib>\Plugins\Front\Plugin.Front.Jetton*
- Пример использования библиотеки - см. iikoCardISSApp.7z

Сценарии настройки

В коде примеров используется вспомогательный класс ServiceCaller. Лучший сценарий использования этого класса:

- создать экземпляр в начале работы приложения;
- установить путь к файлу конфигурации перед первым использованием;
- использовать экземпляр в коде приложения;
- опционально вызвать метод Dispose(), когда вспомогательный класс больше не нужен.

Пример 1

```
public class UserStoryExample
{
private readonly ServiceCaller caller;
public UserStoryExample(string callerConfig)
{
caller = new ServiceCaller();
caller.SetConfigurationFile(callerConfig);
}
}
```

Получить список карт

Получать список всех карт можно при помощи метода GetAccessors().

Пример 2

```
private IEnumerable<CardAccessor> GetCards()
{
// return full version of cards
return caller.Call<IManagementService, IEnumerable<Accessor>>( service =>
service.GetAccessors()).Cast<CardAccessor>();
}
```

Если количество карт большое, то запрос всех данных может потребовать большого времени и больших ресурсов (передача данных, память). Поэтому рекомендуется получать «облегченные» версии объектов при помощи метода GetAllCardAccessorsLight(). Они содержат данные по номеру, датам, но не содержат счетов, владельца и других «сложных» объектов.

Пример 3

```
private IEnumerable<CardAccessor> GetCardsLight()
{
// return light version of cards
return caller.Call<IManagementService, IEnumerable<CardAccessor>>( service =>
```

```
service.GetAllCardAccessorsLight());
}
```

В дальнейшем подробные данные по одной карте можно получить при помощи метода GetCardAccessorById(…).

Пример 4

```
private CardAccessor GetCardInfo(string number)
{
// get card id
Guid addCardId = GetCardsLight().Single(acc => acc.ExternalId == number).Id;
// return full version of card item
return caller.Call<IManagementService, CardAccessor>( service =>
service.GetCardAccessorById(addCardId));
}
```

Добавить карту

Пример 5

```
public CardAccessor AddCard(string groupName, string number)
{
// get group information
AccessorGroup ag =
caller.Call<IManagementService, IEnumerable<AccessorGroup>>( service =>
service.GetAccessorGroups()) .Single(group => group.Name == groupName);
// create card
CardAccessor card = new CardAccessor
{
ExternalId = number,
Number = number,
Activity = ActivityStatus.Active,
Group = ag,
};
// fill counters and accounts by group value
ag.AccessorTemplate.Fill(card, false);
// call service method
caller.Call<IManagementService>(service => service.AddAccessor(card));
return GetCardInfo(number);
}
```

Удалить карту

Пример 6

```
public void RemoveCard(Accessor card)
{
// remove card from server
caller.Call<IManagementService>(service => service.RemoveAccessor(card.Id));
}
```

Обновить карту

Пример 7

```
public CardAccessor UpdateCard(CardAccessor card, string accountName, string counterTypeName,
decimal value, decimal? lowLimit, decimal? highLimit)
{
```

```
AccountCounter accountCounter =
card.Accounts.Single(account => account.Name == accountName).Counters
.Single(counter => counter.CounterType.Name == counterTypeName);
accountCounter.DefaultValue = value;
accountCounter.LimitsValidator.Minimum = lowLimit;
accountCounter.LimitsValidator.Maximum = highLimit;
caller.Call<IManagementService>(service => service.UpdateAccessor(card));
return GetCardInfo(card.Number);
}
```

Сценарии отчетов

Баланс по одной карте

```
Значение по умолчанию.
public decimal GetDefaultValue(string number, string accountName, string counterTypeName)
{
Guid addCardId =
caller.Call<IManagementService, IEnumerable<CardAccessor>>(service =>
service.GetAllCardAccessorsLight())
.Single(acc => acc.ExternalId == number).Id;
return
caller.Call<IManagementService, CardAccessor>(service => service.GetCardAccessorById(addCardId))
.Accounts.Single(account => account.Name == accountName).Counters
.Single(counter => counter.CounterType.Name == counterTypeName).DefaultValue;
}
Баланс.
public decimal GetBalance(string number, string accountName, string counterTypeName)
{
Guid addCardId =
caller.Call<IManagementService, IEnumerable<CardAccessor>>(service =>
service.GetAllCardAccessorsLight())
.Single(acc => acc.ExternalId == number).Id;
return
caller.Call<IManagementService, CardAccessor>(service => service.GetCardAccessorById(addCardId))
.Accounts.Single(account => account.Name == accountName).Counters
.Single(counter => counter.CounterType.Name == counterTypeName).Value;
}
Минимальное ограничение баланса.
public decimal? GetLowLimit(string number, string accountName, string counterTypeName)
{
Guid addCardId =
caller.Call<IManagementService, IEnumerable<CardAccessor>>(service =>
service.GetAllCardAccessorsLight())
.Single(acc => acc.ExternalId == number).Id;
return
caller.Call<IManagementService, CardAccessor>(service => service.GetCardAccessorById(addCardId))
.Accounts.Single(account => account.Name == accountName).Counters
.Single(counter => counter.CounterType.Name == counterTypeName).LimitsValidator.Minimum;
}
```

Баланс по группе карт

Пример 8

```
public IDictionary<string, decimal?> GetBalances(string groupName, string counterName)
{
AccessorGroup ag =
caller.Call<IManagementService, IEnumerable<AccessorGroup>>(
service => service.GetAllAccessorGroupsLight()).Single(group => group.Name == groupName);
CounterType ct =
caller.Call<IManagementService, IEnumerable<CounterType>>(
service => service.GetCounterTypes()).Single(counterType => counterType.Name == counterName);
ConfigurableCardCountersReport reportSettings =
new ConfigurableCardCountersReport
{
AccessorGroups = new [] {ag.Id},
CounterTypes = new [] {ct.Id},
IncludeDeletedCards = false,
};
ConfigurableCardCountersReport report = caller.Call<IReportingService,
ConfigurableCardCountersReport>(
service => service.GetConfigurableCardCountersReport(reportSettings));
return report.Items
.ToDictionary(item => item.Number,
item => item.CounterInfoByCounterType[ct.Id]
.FirstOrDefault(counter => counter.ValueRetrieverType == ValueRetrieverType.SELF_VALUE).Value);
}
```

Доступные средства

Пример 9

```
public AuthorizedClient GetAuthorization(string organization, string card)
{
return caller.Call<IPOSService, AuthorizedClient>( service => service.RequestAuthorization(organization,
card));
}
```

Отчет по операциям

Пример 10

```
public IList<DetailedCardUsageReportItem> GetOperationsListReport(string groupName, string
serviceName, DateTime from, DateTime to)
{
AccessorGroup ag =
caller.Call<IManagementService, IEnumerable<AccessorGroup>> (
service => service.GetAllAccessorGroupsLight()).Single(group => group.Name == groupName);
Service s =
caller.Call<IManagementService, IEnumerable<Service>> (
service => service.GetServices()).Single(service => service.Name == serviceName);
DetailedCardUsageReport report = new DetailedCardUsageReport
{
CardGroupsFilter = new[] {ag.Id},
From = from,
To = to,
ServicesFilter = new[] { s.Id },
```

```
IncludeDeletedCards = true,
};
var result = caller.Call<IReportingService, DetailedCardUsageReport> (service =>
service.GetDetailedCardUsagesReport(report));
return result.Items;
}
```

В DetailedCardUsageReportItem поле TotalValue означает сумму, на которую изменился баланс карты в результате операции. Т.е. в случае оплаты картой это поле будет отрицательным. Поле RequestId – уникальный идентификатор операции в системе. При построении отчета даты from и to учитываются полностью, т.е. фильтрация идет по датам с учетом часов, минут, секунд и миллисекунд.

Сценарии операций

Выполнить операцию

```
public void DoOperation(Guid id, string organization, string card, string operation, decimal value)
{
SafeServiceCaller caller = new SafeServiceCaller();
caller.Call<IPOSService>(service => service.ExecuteById(id, organization, card, operation, value));
}
```

Параметры метода:

- id – идентификатор операции
- organization – идентификатор организации
- card – идентификатор карты
- operation – идентификатор операции
- value – сумма операции