API Link:

(POST, GET):

[FlexibilityOfferFunction: https://flexchain-functionapp-testenv.azurewebsites.net/api/flexibilityOffers](FlexibilityOfferFunction:%20https://flexchain-functionapp-testenv.azurewebsites.net/api/flexibilityOffers)

FlexibilityRequestFunction: <https://flexchain-functionapp-testenv.azurewebsites.net/api/flexibilityRequests>

Flex\_MatchingAlgo\_Function: <https://flexchain-functionapp-testenv.azurewebsites.net/api/flex_matching_algo_Results>

(GET) (a specific entry)

[FlexibilityOfferGetFunction: https://flexchain-functionapp-testenv.azurewebsites.net/api/flexibilityOffers](FlexibilityOfferGetFunction:%20https://flexchain-functionapp-testenv.azurewebsites.net/api/flexibilityOffers)/{UserID}

[FlexibilityRequestGetFunction: https://flexchain-functionapp-testenv.azurewebsites.net/api/flexibilityRequests](FlexibilityRequestGetFunction:%20https://flexchain-functionapp-testenv.azurewebsites.net/api/flexibilityRequests)/{RequestID}

Flex\_MatchingAlgo\_Get\_Function: <https://flexchain-functionapp-testenv.azurewebsites.net/api/flex_matching_algo_Results>/{RequestID}

(DELETE: specific entry) [by DSO: ALL USING {RequestID}]

[FlexibilityOfferDelFunction: https://flexchain-functionapp-testenv.azurewebsites.net/api/flexibilityOffers](FlexibilityOfferDelAllFunction:%20https://flexchain-functionapp-testenv.azurewebsites.net/api/flexibilityOffers)/{RequestID}

[FlexibilityRequestDelFunction: https://flexchain-functionapp-testenv.azurewebsites.net/api/flexibilityRequests](FlexibilityRequestGetFunction:%20https://flexchain-functionapp-testenv.azurewebsites.net/api/flexibilityRequests)/{RequestID}

Flex\_MatchingAlgoDelFunction: <https://flexchain-functionapp-testenv.azurewebsites.net/api/flex_matching_algo_Results>/{RequestID}

(Blind MatchingAlgo GET):

Flex\_MatchingAlgo\_Blind\_Function: [http://localhost:7071/api/flex\_matching\_algo\_Results\_blind/{RequestID}](http://localhost:7071/api/flex_matching_algo_Results_blind/%7bRequestID%7d)

Validity Check Function (GET):

ValidityCheck\_Get\_Function: http://localhost:7071/api/flex\_validity\_check/

**Offer Format:**

"UserId": "user001",

            "FlexOfferList": [

                {

                    "BidPriceCtpEUList": "null",

                    "endFlexShiftTimeSlot": "2021-10-05T21:00:00Z",

                    "RequestId": "Req000001",

                    "startFlexShiftTimeSlot": "2021-10-05T19:00:00Z",

                    "totalFlexOfferedEU": -6,

                    "id": **null**

                },

                {

                    "BidPriceCtpEUList": "null",

                    "endFlexShiftTimeSlot": "2021-10-05T19:15:00Z",

                    "RequestId": "Req000002",

                    "startFlexShiftTimeSlot": "2021-10-05T19:00:00Z",

                    "totalFlexOfferedEU": 2,

                    "id": **null**

                },

                {

                    "BidPriceCtpEUList": "8, 10, 12, 13",

                    "endFlexShiftTimeSlot": "2021-10-05T19:15:00Z",

                    "RequestId": "Req000006",

                    "startFlexShiftTimeSlot": "2021-10-05T19:00:00Z",

                    "totalFlexOfferedEU": 4,

                    "id": **null**

                },

                {

                    "BidPriceCtpEUList": "7",

                    "endFlexShiftTimeSlot": "2021-10-05T19:15:00Z",

                    "RequestId": "Req000007",

                    "startFlexShiftTimeSlot": "2021-10-05T19:00:00Z",

                    "totalFlexOfferedEU": 4,

                    "id": **null**

                }

            ]

**Request Format:**

**(for fixed price):**

"RequestId": "Req000001",

            "Mode": "fcfs",

            "FullfillmentFactor": "50",

            "IfFlexRequested": **true**,

            "Loc": {

                "user001": 3,

                "user002": 2,

                "user003": 5,

                "user004": 4

            },

            "MarketType": "fixedPrice",

            "MaxPriceCtpEU": "null",

            "PriceOfferCtpEU": "7",

            "ReferencePriceCtpEU": "null",

            "TimeSlot": "2021-10-05T15:15:00Z",

            "TotalFlexRequestedEU": "-11",

“MatchingAlgoCheck”: false

 "RequestId": "Req000005",

            "Mode": "fcfs",

            "FullfillmentFactor": "0",

            "IfFlexRequested": **false**,

            "loc": {

                "null": 0

            },

            "MarketType": "fixedPrice",

            "MaxPriceCtpEU": "null",

            "PriceOfferCtpEU": "null",

            "ReferencePriceCtpEU": "null",

            "TimeSlot": "2021-06-13T22:16:00Z",

            "TotalFlexRequestedEU": "null",

“MatchingAlgoCheck”: false

**(for auction):**

 "RequestId": "Req000006",

            "Mode": "MiP",

            "FullfillmentFactor": "50",

            "IfFlexRequested": **true**,

            "loc": {

                "user001": 1,

                "user002": 2,

                "user003": 5,

                "user004": 1

            },

            "MarketType": "auction",

            "MaxPriceCtpEU": "12",

            "PriceOfferCtpEU": "null",

            "ReferencePriceCtpEU": "10",

            "TimeSlot": "2021-10-05T15:00:00Z",

            "TotalFlexRequestedEU": "7",

“MatchingAlgoCheck”: false

 "RequestId": "Req000008",

            "Mode": "MiP",

            "FullfillmentFactor": "null",

            "IfFlexRequested": **false**,

            "loc": {

                "null": 0

            },

            "MarketType": "auction",

            "MaxPriceCtpEU": "null",

            "PriceOfferCtpEU": "null",

            "ReferencePriceCtpEU": "null",

            "TimeSlot": "2021-10-05T15:00:00Z",

            "TotalFlexRequestedEU": "null",

“MatchingAlgoCheck”: false

**Details on the Modes:**

Types of modes:

fcfs, maah, maav, mapw, miah, miav, mip, zufall

**FCFS:**

If flexibilityOffer.UserId is user

{

For each uservalue in flexibilityOffer.FlexOfferList

{

If uservalue.RequestID is flexibilityRequest.RequestID

{

If SIGN>0 && uservalue.TotalFlexOfferedEU > 0) or (SIGN<0 && uservalue.TotalFlexOfferedEU < 0

{

If SIGN<0 && uservalue.TotalFlexOfferedEU < 0

{

uservalue.TotalFlexOfferedEU = Math.Abs(uservalue.TotalFlexOfferedEU)

}

If TOTALFLEXREQUESTED - uservalue.TotalFlexOfferedEU > 0

{

TOTALFLEXREQUESTED-= uservalue.TotalFlexOfferedEU

accepted\_offers[flexibilityOffer.UserId] = uservalue.TotalFlexOfferedEU

}

else

{

accepted\_offers[flexibilityOffer.UserId] = TOTALFLEXREQUESTED

return accepted\_offers;

}

}

}

}

}

MAAH:

For each uservalue in flexibilityOffer.FlexOfferList

{

if uservalue.RequestID is flexibilityRequest.RequestID

{

if ((SIGN > 0 && uservalue.TotalFlexOfferedEU > 0) or (SIGN < 0 && uservalue.TotalFlexOfferedEU < 0))

{

if SIGN < 0 && uservalue.TotalFlexOfferedEU < 0)

{

uservalue.TotalFlexOfferedEU = Math.Abs(uservalue.TotalFlexOfferedEU)

}

POTENTIALOFFER[flexibilityOffer.UserId] = uservalue.TotalFlexOfferedEU

}

}

}

SUM=0

For each i in POTENTIALOFFER.Keys

{

SUM += POTENTIALOFFER[i];

}

if SUM <= TOTALFLEXREQUESTED

{

return POTENTIALOFFER

}

else

{

var ordered = POTENTIALOFFER.OrderBy(x.Value).todictionary();

for each i in ordered.Keys

{

if TOTALFLEXREQUESTED > ordered[i]

{

accepted\_offers[i] = ordered[i]

TOTALFLEXREQUESTED -= ordered[i]

}

else

{

accepted\_offers[i] = TOTALFLEXREQUESTED

return accepted\_offers

}

}

return accepted\_offers;

}

}

MAAV:

if uservalue.RequestID is flexibilityRequest.RequestID

{

if ((SIGN > 0 && uservalue.TotalFlexOfferedEU > 0) or (SIGN < 0 && uservalue.TotalFlexOfferedEU < 0))

{

if (SIGN < 0 && uservalue.TotalFlexOfferedEU < 0)

{

uservalue.TotalFlexOfferedEU = Math.Abs(uservalue.TotalFlexOfferedEU)

}

var usertotalflexrequested = uservalue.TotalFlexOfferedEU

while (usertotalflexrequested not 0)

{

POTENTIALOFFER.Add(flexibilityOffer.UserId)

usertotalflexrequested--

}

}

}

POTENTIALOFFER.Shuffle();

if POTENTIALOFFER.Count() > TOTALFLEXREQUESTED

{

while(TOTALFLEXREQUESTED not 0)

{

if(accepted\_offers.ContainsKey(POTENTIALOFFER[0]))

{

accepted\_offers[POTENTIALOFFER[0]] = accepted\_offers[POTENTIALOFFER[0]] + 1

}

else

{

accepted\_offers[POTENTIALOFFER[0]] = 1

}

TOTALFLEXREQUESTED--

POTENTIALOFFER.RemoveAt(0)

}

}

else

{

For each offer in POTENTIALOFFER

{

If accepted\_offers.ContainsKey(offer)

{

accepted\_offers[offer] = accepted\_offers[offer] + 1

}

else

{

accepted\_offers[offer] = 1

}

}

}

return accepted\_offers;

}

MAPW:

if (uservalue.RequestID == flexibilityRequest.RequestID)

{

if ((SIGN > 0 && uservalue.TotalFlexOfferedEU > 0) || (SIGN < 0 && uservalue.TotalFlexOfferedEU < 0))

{

if (SIGN < 0 && uservalue.TotalFlexOfferedEU < 0)

{

uservalue.TotalFlexOfferedEU = Math.Abs(uservalue.TotalFlexOfferedEU);

}

POTENTIALOFFER[flexibilityOffer.UserId] = uservalue.TotalFlexOfferedEU;

}

}

var SUM = 0;

for each i in POTENTIALOFFER.Keys

{

SUM+= POTENTIALOFFER[i]

}

If SUM<= TOTALFLEXREQUESTED

{

return POTENTIALOFFER

}

else

{

var PEFACTOR = new Dictionary<string, int>()

PEFACTOR = flexibilityRequest.Loc

PEFACTOR = PEFACTOR.OrderByDescending(x => x.Value).todictionary()

For each i in PEFACTOR.Keys

{

If TOTALFLEXREQUESTED> POTENTIALOFFER[i]

{

accepted\_offers[i] = POTENTIALOFFER[i]

TOTALFLEXREQUESTED-= POTENTIALOFFER[i]

}

else

{

accepted\_offers[i] = TOTALFLEXREQUESTED

return accepted\_offers

}

}

}

return accepted\_offers;

MIAV:

if uservalue.RequestID is flexibilityRequest.RequestID

{

if ((SIGN > 0 && uservalue.TotalFlexOfferedEU > 0) or (SIGN < 0 && uservalue.TotalFlexOfferedEU < 0))

{

if SIGN < 0 && uservalue.TotalFlexOfferedEU < 0

{

uservalue.TotalFlexOfferedEU = Math.Abs(uservalue.TotalFlexOfferedEU)

}

POTENTIALOFFER[flexibilityOffer.UserId] = uservalue.TotalFlexOfferedEU

}

}

For each i in POTENTIALOFFER.Keys

{

SUM += POTENTIALOFFER[i]

}

if (SUM <= TOTALFLEXREQUESTED)

{

return POTENTIALOFFER

}

else

{

var ordered = POTENTIALOFFER.OrderByDescending(x => x.Value).todictionary()

for each i in ordered.Keys

{

if (TOTALFLEXREQUESTED > ordered[i])

{

accepted\_offers[i] = ordered[i]

TOTALFLEXREQUESTED -= ordered[i]

}

else

{

accepted\_offers[i] = TOTALFLEXREQUESTED

return accepted\_offers

}

}

return accepted\_offers

}

MIAH:

The logic for this trading mode is identical with “Minimale Anzahl Verträge (MiAV)”

MiP:

if (uservalue.RequestID is flexibilityRequest.RequestID)

{

if ((SIGN > 0 && uservalue.TotalFlexOfferedEU > 0) or (SIGN < 0 && uservalue.TotalFlexOfferedEU < 0))

{

if (SIGN < 0 && uservalue.TotalFlexOfferedEU < 0)

{

uservalue.TotalFlexOfferedEU = Math.Abs(uservalue.TotalFlexOfferedEU);

}

BIDS[user] = uservalue.BidPriceCtpEUList;

}

}

For each username in BIDS.Keys

{

var COUNT = 0

var userbids = BIDS[username]

for each userbid in userbids

{

if userbid <= flexibilityRequest.MaxPriceCtpEU

{

POTENTIALOFFER[ (COUNT).ToString() + " " + username] = userbid

COUNT+=1;

}

}

}

var sortedDict = POTENTIALOFFER.OrderBy(x => x.Value).todictionary()

for each key in sortedDict.Keys

{

if(TOTALFLEXREQUESTED>1)

{

TOTALFLEXREQUESTED--;

if (accepted\_offers.ContainsKey(key))

{

accepted\_offers[key] += POTENTIALOFFER[key];

}

else

{

accepted\_offers[key] = POTENTIALOFFER[key];

}

}

else

{

if (accepted\_offers.ContainsKey(key))

{

accepted\_offers[key] += POTENTIALOFFER[key];

}

else

{

accepted\_offers[key] = POTENTIALOFFER[key];

}

return accepted\_offers

}

}

return accepted\_offers

Zufall:

trading mode is identical with “Maximal Anzahl Verträge (MaAV)”

At one time all the request modes should be same (all fcfs, or all mip or all maav …).

**Note:**

**For** <https://flexchain-functionapp-testenv.azurewebsites.net/api/flex_matching_algo_Results>

**We do not put any details (Json) as the body for POST request. The server will take data from FlexibilityRequest and FlexibilityOffer and use the Matching Algorithm. The result of the matching algorithm is stored in the the flex\_matching\_algo\_Results table in the database.**

Matching Algo Result Format (The format in which matching algo results are stored in the table):

{

            "Requests": {

                "Req000001": {

                    "user001": 3,

                    "user003": 2,

                    "user004": 2,

                    "user002": 4

                },

                "Req000002": {

                    "fullfillment factor did not reach": 0

                },

                "Req000003": {

                    "ifflexrequested false": 0

                },

                "Req000004": {

                    "ifflexrequested false": 0

                },

                "Req000005": {

                    "ifflexrequested false": 0

                }

            },

            "id": "62b59a3b3b731ca98e88791c"

        },

        {

            "Requests": {

                "Req000001": {

                    "user001": 4,

                    "user002": 6,

                    "user003": 1

                },

                "Req000002": {

                    "fullfillment factor did not reach": 0

                },

                "Req000003": {

                    "ifflexrequested false": 0

                },

                "Req000004": {

                    "ifflexrequested false": 0

                },

                "Req000005": {

                    "ifflexrequested false": 0

                }

            },

            "id": "62beac901730b8d0d7eb7ed5"

        }

**DeleteALL** Cron Job:

Every 2 weeks.

await requestcontroller.DropCollection();

await offerController.DropCollection();

await matchingController.DropCollection();

**MatchingAlgo POST** Cron Job:

Every 15 min.

await matchingController.Post(null);

**(Blind MatchingAlgo GET):**

To call [http://localhost:7071/api/flex\_matching\_algo\_Results\_blind/{RequestID}](http://localhost:7071/api/flex_matching_algo_Results_blind/%7bRequestID%7d)

As GET method.

Body:

**{**

**“userId” : “value”,**

**“password” : “value”**

**}**