Référentiel technique Systèmes de pilotage énergétique Flex Ready

Version	Date	Auteurs	Description
1.0	14/04/2025	TSG	Création du fichier Référentiel technique Systèmes de pilotage énergétique Flex Ready

Table des matières

1. Cibles du référentiel Systèmes de pilotage énergétique Flex Ready®

La compatibilité au référentiel Systèmes de Pilotage Energétique Flex Ready® n'est possible que pour :

- Les systèmes de pilotage énergétique répondant aux critères techniques fonctionnels décrits ci-dessous.
    - **NB :** le cadre de référence Flex Ready version pilote ne spécifie que les systèmes de pilotage automatisés répondant au cadre de référence et non les capteurs ou actionneurs en aval.
- Les fournisseurs d'électricité intégrant les requêtes fournisseurs (cf. 2.2.1) de l'API Flex Ready® en vigueur et pouvant ainsi s'interfacer avec les systèmes de pilotage énergétique.
- Les opérateurs d'effacemet intégrant les requêtes opérateurs d'effacement de niveau 1de l'API Flex Ready® (cf. 2.2.2) en vigueur et pouvant ainsi s'interfacer avec les systèmes de pilotage énergétique.

Le droit d'usage s'adresse aux personnes morales souhaitant apposer la marque Flex Ready® sur leurs produits et/ou services.

2. Prérequis techniques

2.1. Prérequis techniques fonctionnels

REGULATION	**Généraux**	Capacité à recevoir une consigne en puissance et la transformer en modification de consigne (exemple -2°) pour les usages en aval.
	Chauffage	Régulation centralisée automatique avec capteurs*
	Climatisation et autres systèmes de refroidissement	Régulation centralisé automatique avec capteurs
	Ventilation	Régulation automatique et programmation centralisées
	Eau chaude sanitaire	Régulation automatique avec programme quotidien
	IRVE	Commande marche arrêt et modulation de l'IRVE
	Stockage	Capacité à piloter un système de stockage électrochimique et/ou thermique
GESTION DE L'ENERGIE ET SUPERVISION	**Gestion technique centralisée**	Gestion des points de consigne de tous les équipements relié
		Gestion des plages horaires (minimum 15 minutes)
		Gestion des défauts et des alarmes (Supervision)
	Données énergétiques (EMS)	Suivi, enregistrement et analyse des données réalisé par zone fonctionnelle
		Suivi, enregistrement et analyse des données réalisé en continu et à pas de temps 15 minutes

		Données énergétiques conservées et accessibles pendant 5 ans
		Adaptation des scénarios du système en fonction d'un signal prix
		Interopérabilité (extraction des données de la GTC)
COMMUNICATION FLEX READY	**Echange des donnés de flexibilité**	Accès à internet
		Intégration de l'API Flex Ready (spécifications GIMELEC en annexe)
		Intégration de l'API EcoWatt (lien vers RTE)
		Intégration du socle de référence cyber BACS Flex Ready (spécifications GIMELEC en annexe)
		En option : Intégration des fonctionnalités vers le compteur ENEDIS (spécifications en annexe/lien vers)

2.2. Spécification de l'API Flex Ready®

Le but de l'API Flex Ready® est de permettre :

- À un fournisseur d'électricité de transmettre des informations tarifaires vers un bâtiment équipé d'un système de pilotage énergétique
- À un opérateur de flexibilité (aussi appelé agrégateur dans ce document) et à un système de pilotage énergétique d'échanger des requêtes de flexibilité et les réponses à ces requêtes, de manière standardisée

L'API intègre les éléments de la norme IEC 62746-4[1] qui sont pertinents sur le cas d'usage. La norme IEC 62746-4 définit (annexe 2) la séquence d'échanges entre les acteurs pour le *Use case: Incentive-based building energy management*. L'API Flex Ready® se base sur cette séquence pour les échanges qu'elle traite, concrètement cette API traite les uses cases :

- Flexibilité implicite : Ajustement/optimisation des consommations électriques dans un bâtiment en réception des signaux tarifaires.
- Flexibilité explicite : Ajustement/effacement des consommations électriques dans un bâtiment en réception d'un signal extérieur d'un acteur tiers, UC flexibilité explicite hors services temps réel.

Cette partie 3.2 décrit l'ensemble des fonctions et requêtes de l'API et précise, pour chacune, ce qui doit être implémenté pour être conforme au référentiel Flex Ready® v1.

L'annexe 1 comporte la description complète des requêtes et décrit le fonctionnement des scénarios de test de compatibilité via l'émulateur.

D'une façon générale, le système de pilotage énergétique dispose d'un ID_bat sur la base de l'ID_gam de la gamme définie lors de l'enregistrement Flex Ready® (cf. 2.2.2). Le système de pilotage énergétique contrôle

[1] Disponible ici

des Assets, qui peuvent correspondre à un périmètre fonctionnel (Chauffage, IRVE…) ou à des zones du bâtiment, voire le bâtiment entier.

En complément, les Assets ont chacun un identifiant défini par le gestionnaire du système de pilotage énergétique et communiqué à l'opérateur de flexibilité ou le fournisseur d'électricité dans le contrat. Ces identifiants peuvent être identiques entre 2 systèmes de pilotage énergétique (ils ne sont pas gérés par Flex Ready®). Différents Assets peuvent être gérés par différents opérateurs de flexibilité : l'API est ouverte à cette possibilité. Les API sont sécurisées, chaque acteur étant responsable et maître du choix cyber.

2.2.1. Requêtes à disposition des fournisseurs de l'API Flex Ready®

Les requêtes de l'API Flex Ready® à disposition des fournisseurs ont pour but de transmettre des informations entre un fournisseur d'électricité vers un système de pilotage énergétique.

Les requêtes fournisseurs sont uniquement accessibles en lecture, via une opération de type GET du gestionnaire de bâtiment, avec son identifiant unique en paramètre d'entrée.

Dans la version 1, l'API intègre des requêtes permettant à un système de pilotage énergétique de recevoir des informations sur l'évolution de son contrat de fourniture pour un intervalle de temps et sur un pas de temps défini lors de la requête.

Basée sur la norme IEC 62746-4, l'API a vocation à couvrir des produits de flexibilité implicite plus ou moins dynamiques allant de tarifs bases avec heures pleines heures creuses classiques à des offres de fournitures indexées tout ou en partie sur le SPOT.

- Demande information tarifaire :

GetCommodityPriceExchange

Le système de pilotage énergétique interroge son fournisseur sur les tarifs de fourniture de son contrat avec comme donnée d'entrée :

- DateTimeInterval : en heures
- PDL

Envoie trois informations pour les X heures glissantes (DateTimeInterval) à partir de l'heure de l'interrogation, décomposé à un pas de 15 minutes :

- puissance souscrite ;
- prix du kWh ;
- empreinte carbone de l'électricité (facultatif, peut être vide)

Chaque identifiant possède un quota d'appels autorisés qu'il peut effectuer dans un intervalle de temps donné. La valeur du quota est d'un appel toutes les 15 minutes pour cette ressource. Un code erreur est transmis en cas de tentatives d'interrogation trop rapprochées. Etant donné la faible variabilité des informations remontées, il est recommandé d'effectuer un requêtage une fois par jour.

L'accès à l'API se fait via le format d'API REST et utilise le format de fichier JSON. L'API est uniquement accessible en lecture, via une opération de type GET du gestionnaire de bâtiment, avec son ID_bat unique (cf. 2.2.2) en point d'entrée.

La réponse est décomposée en pas de 15 minutes, chaque interrogation retourne donc DateTimeInterval*4 pas pour couvrir les X heures à venir. L'API retourne également la date et l'heure de chaque pas afin de permettre une meilleure lisibilité des informations.

SupplierSignal			Tableau de valeurs {JSON} structuré comme suit :			
	Champ		**Card.**	**Type**	**Description**	**Valeurs / Format**
DateTimeInterval	**FileGeneration**		1..1	date	Date de dernière génération des données	YYYY-MM-DDThh:mm:ss+02:00
	DP		1..1	numérique	PDL du bâtiment	Valeur numérique 14 chiffres
	DateTimeInterval		1..1	numérique	Durée en heures à partir de l'heure de requête	Valeur numérique
	Values		Une valeur par pas Horaire de 0 à X structuré comme suit :			
	15	**Step**	1..1	numérique	Pas horaire de 1 à X*4	1 puis 2… jusqu'à X*4 pour les X heures à venir
		Day	1..1	date	Date et heure du pas de temps concerné	YYYY-MM-DDThh:mm:ss+02:00
		Power	1..1	numérique	Puissance souscrite en kVa	Valeur numérique
		Cost	1..1	numérique	Coût en € par kWh	Valeur numérique
		CO2	1..1	numérique	Quantité de CO2 émise en gCO2eq/kWh ou autre information plus pertinente	Valeur numérique

2.2.2. Requêtes à disposition des opérateurs d'effacement

2 paramètres principaux sont à prendre en compte :

- la capacité du BACS d'établir une prévision de ses consommations futures (horizon H + 2 à J + 2), potentiellement par usage
- le choix du gestionnaire du bâtiment de s'engager sur un niveau de flexibilité.

Trois niveaux de BACS considérés :

- Niveau 1 : BACS sans capacité de prévision – pas d'engagement contractuel de flexibilité
- Niveau 2: BACS sans capacité de prévision, avec engagement contractuel de flexibilité

Les API sont cumulatives (niveau 2: le niveau 1 plus le spécifique du niveau 2).

Les API sont sécurisées, selon les exigences de cyber sécurité définies dans un document séparé.

La norme IEC 62746-4 définit dans l'annexe A la séquence d'échanges entre les acteurs pour le Use case: Incentive-based building energy management L'API flexREADY se base sur cette séquence (jointe en fin de document) pour les échanges qu'elle traite.

Data model :

Autant que possible, le Data model utilisé est celui de la norme 62746-4.

D'une façon générale, le BACS dispose d'un identifiant BACSID, attribué lors de l'enregistrement FlexReady®. Cet identifiant doit être unique (UUID – cf norme 62746-4 §5.1 Master resource identifiers).

Le BACS contrôle des Assets, qui peuvent correspondre à un périmètre fonctionnel (Chauffage, IRVE…) ou à des zones du bâtiment, voire le bâtiment entier.

Le nom d'asset « WholeBuilding » est réservé pour le cas où le seul asset pris en compte est le bâtiment entier. Dans ce cas, aucun autre assetID n'est défini.

Les identifiants AssetID des Assets sont définis par le gestionnaire du BACS. Ils sont communiqués à l'opérateur dans le contrat. Ces identifiants peuvent être identiques entre 2 BACS (ils ne sont pas gérés par FlexReady). Différents Assets peuvent être gérés par différents opérateurs de flexibilité : l'API est ouverte à cette possibilité.

L'heure est exprimée en heure GMT. Les dates sont au format ISO 8601 : Ainsi, pour les jours de la semaine : 1: lundi, 2 : mardi, … 7 : dimanche

Conformément à la norme IEC 62746-4, FloatQuantity est une valeur accompagnée d'une unité et d'un multiplicateur (k, M, …)

Une PowerTimeSeries de puissance est une séquence [puissance: FloatQuantity, date: Date, debut: Heure, fin: Heure]. Cette séquence peut n'être constituée que d'un seul point.

Une requête de flexibilité correspond à un FlexProductType. Dans cette version de l'API, 2 types sont utilisés, selon l'horizon de temps de la requête:

- WID : Wholesale IntraDay Market
- WDA : WholesaleDay ahead Market

La version européenne de la norme prévoit l'utilisation de TimeSeries. Cette notion est donc introduite dans la spécification, la série pouvant être limitée à 1 seul point.

Certaines API retournent un code d'erreur :

- 200 OK : Succès
- 202 Accepted : Demande acceptée avec succès
- 400 Bad Request : Erreur de format dans la requête
- 422 Unprocessable Entity : Erreur de validation métier

A ce stade, les API n'autorisent pas de données supplémentaires ; une analyse de l'impact cyber sécurité sera faite pour définir dans quelles conditions des variables supplémentaires pourraient être introduites par l'utilisateur.

2.2.2.1. API Niveau 1

Le BACS n'a pas la capacité d'estimer sa consommation à venir. Le bâtiment ne peut donc s'engager contractuellement avec l'opérateur de flexibilité, mais simplement de faire ses best efforts.

• Potentiel des assets du bâtiment : l'opérateur interroge le serveur du bâtiment ; l'API fournit une information statique à l'opérateur, sur le potentiel théorique des assets qui peuvent entrer dans la flexibilité (qui peut être évaluée selon les méthodes GOFlex).

func getBACSAssets(bacs: BACSID) → [(asset: AssetID, flexProduct : FlexProductType, potentiel:

Protocole : Timeout : 60s (équivalent à celui d'une requête de page web). En cas de non réponse, un message d'erreur est généré et loggé. Si le BACSID est erroné, le BACS retourne une liste vide (besoin de code erreur?).

L'API fournit un tableau donnant pour chaque asset un potentiel de flex, en indiquant pour quel type de service de flexibilité.

potentiel est un tableau de PotentielFlex (plusieurs éléments si on distingue en fonction de la saisonnalité par exemple). Dans la version 1, une seule valeur de potentiel, à l'instant de la requête. Le niveau de confiance n'est pas exploité en V1.

PotentielFlex est une structure de données contenant : la puissance, le délai de mobilisation (par défaut: 1 h), la durée max, le nombre d'activations dans la journée (par défaut 1, seule valeur acceptée en V1), les périodes horaires, –optionnellement un niveau de confiance sur la mise en œuvre (exclu en V1). Le potentiel est celui valable à date.

PotentielFlex :

yearPeriod : startDay, endDay
activationPeriods: (months : [1..12], weekDay : [1..7], hours: [Hour]) inside the yearPeriod
notification : Int Une durée exprimée en minutes
maxDuration : Int Une durée exprimée en minutes, depuis le début d'une période d'activation
maxActivationsPerDay : Int
confidence level : 1..5 1 minimum – 5 confiance totale

Par exemple :

yearPeriod : (2025-03-25, 2025-06-30)

ActivationPeriods : (months : [3, 4, 6], weekDay : [6,7], hours: [10, 17]) : la flex peut être activée (sur cette période de l'année, tous les mois sauf Mai, les samedis et dimanche, à 10h GMT et 17h GMT, pour une durée maxDuration

Les différentes yearPeriod doivent être cohérentes. En cas d'incohérence (par ex : un instant donné apparaissant dans 2 PotentielFlex), le résultat n'est pas garant i (en général, le premier qui correspond à la période sera pris en compte).

Un assetID est prédéfini pour le bâtiment complet. Si cet asset existe, la liste des assets en réponse ne doit contenir que seul asset.

• Consigne envoyée au BACS : L'opérateur envoie une demande au BACS (et génère un identifiant pour la requête) pour une période donnée, par asset (l'Asset peut être le bâtiment complet); mais sans avoir de garantie de respect. Pas de réponse attendue excepté un accusé de réception (ou notification d'erreur dans les données) qui permet à l'opérateur de relancer (1 fois en V1) sa requête en cas de non réponse ou d'erreur.
Ack indique la bonne réception et le fait que le BACS ne s'engage pas (car niveau 1) : recuSansEngagement.

func askFlex(requestID: String, bacs: BACSID, asset: AssetID, flexProduct : FlexProductType, puissance:

Protocole : Timeout : 5s. Le BACS n'a pas d'évaluation à faire, il indique seulement que la demande a été reçue, afin de permettre à l'opérateur d'intégrer cette requête dans ses calculs de potentiel de flex.
Le BACS a pu vérifier que la demande est cohérente avec le potentiel de flex. Sinon, ne répond pas, ce qui annule la requête.

• Confirmation envoyée au BACS : L'opérateur envoie au BACS qu'il a retenu dans son plan (en fonction de son analyse de probabilité de mise en œuvre par le bâtiment) la confirmation de la demande. La requête est envoyée en tenant compte du délai de mobilisation. Le BACS n'a pas d'engagement d'exécution. Pas de réponse attendue excepté un accusé de réception. Le requestID (UUID généré par l'émetteur) permet de garder une référence de la requête.

func confirmFlexRequest(requestID: String, bacs: BACSID, asset: AssetID, flexProduct :

Protocole : Timeout : 5s. Le BACS n'a pas d'évaluation à faire, il indique seulement que la demande est enregistrée et qu'il fera ses best efforts pour la satisfaire (l'incentive étant la rémunération prévue dans le contrat). En cas de non réponse, l'agrégateur exclue cette requête de ses potentiels de flex.

• Annulation requête : absent en V1

func cancelFlexRequest(requestID: String) → Ack PowerTimeSeries)]

• Reporting du réalisé : L'opérateur demande au BACS la courbe de puissance appelée (à un pas de temps de 15') sur la période de la requête de flexibilité, pour l'Asset considéré. Le BACS répond par un tableau donnant, au pas d'échantillonnage retenu (15'), la puissance appelée par Asset (note : bacs: BACSID, asset: AssetID rappelés par sécurité, mais normalement redondant avec requestID. La définition du calcul du réalisé est à préciser dans le contrat. En V1: valeur moyenne (donc energie) de la puissance sur la période.

func realiseFlexRequest(requestID: String, bacs: BACSID, asset: AssetID) → [(requestID: String, reported :

Protocole : Timeout : 15'. Le BACS doit collecter des données (éventuellement lecture du compteur), ce qui justifie le timeout important. En cas de non réponse, l'agrégateur répète au maximum 1 fois sa requête puis génère un message d'erreur pour log. Si l'opérateur détecte une puissance > potentiel de flex de l'asset : générer erreur dans le log.

2.2.2.1. API Niveau 2

Le BACS n'a pas la capacité d'estimer sa consommation ni donc sa flexibilité à venir. Le bâtiment délègue cette prévision à l'opérateur pour s'engager contractuellement avec l'opérateur de flexibilité.

Au niveaux 2 et 3, une proposition de flexibilité peut être accompagnée d'une proposition de prix. Le BACS acceptera la proposition ou la rejettera en tenant éventuellement compte de cette proposition.

• Potentiel des assets du bâtiment : idem niveau 1

• Consommation instantanée par asset (moins d'une heure d'ancienneté) des assets qui peuvent entrer dans la flexibilité, à l'heure de la requête

func getBACSAssetsConsos(bacs: BACSID) → [(asset: AssetID, date: Date, heure: Heure, conso:

Protocole : Timeout : 15'. Le BACS doit collecter des données (éventuellement interrogation des assets), ce qui justifie le timeout important. En cas de non réponse, l'agrégateur répète au maximum 1 fois sa requête puis génère un message d'erreur pour log.

• Consommation historique des assets.
Le BACS a archivé ses consommations sur une durée d'historique à définir (dans le contrat, minimum 1 mois), en tant que valeur moyenne (ou pMax ultérieurement) sur un pas de temps (1 heure) défini dans le contrat. L'opérateur peut récupérer cet historique (par asset), ce qui lui donne une information très importante pour construire sa prévision:

func getBACSAssetsHisto(bacs: BACSID, asset: AssetID) → [(date: Date, heure: Heure, conso:

Protocole : Timeout : 60s. En cas de non réponse, un message d'erreur est généré et loggé. Si le BACSID ou l'AssetID est erroné, le BACS retourne une liste vide (besoin de code erreur?).
Un historique de 1 mois correspond à 720 valeurs par asset.

• L'opérateur envoie une proposition au BACS avec offre de prix, qui l'accepte ou la refuse ou demande de renégocier dans sa Reponse.

func askFlexWithPrice(bacs: BACSID, asset: AssetID, flexProduct : FlexProductType, puissance:

La réponse est explicite (acceptée, refusé, à modifier), ce qui différencie de l'Ack du niveau 1.
Protocole : Timeout : 15'. Le BACS doit évaluer sa capacité à exécuter la demande de flex, ce qui justifie la durée du timeout. Si le BACS ne sait pas évaluer, il doit refuser la demande pour cet asset. Si un identifiant est incorrect, le BACS répond par un code erreur (identifiant inconnu) ou laisse expirer le timeout.
En cas de non réponse, la requête peut être répétée 1 fois.
Si le BACS refuse, la requête n'est pas réémise.
Dans la version 1 du protocole, la modification ne peut porter que sur la puissance. Si le prix ne convient pas, réponse de refus.

Note: 62746-4 définit ProductBid as « a container which has two associations: the top being the MarketProduct (Energy, Reserve, Regulation, etc.) and the bottom association being the BidPriceSchedule ».
The BidPriceSchedule class captures data on three axes signifying:

1) The time window, e.g. from 6:00 to 7:00
2) The amount of service, e.g. 1 MWh
3) The asking price, e.g. 20 €/MWh

Dans le cas de l'API Flex, le MarketProduct est une demande de flexibilité (Regulation up or down).
Les paramètres Puissance, date, debut, fin, askingPrice pourront être regroupés dans un BidPriceSchedule.
Cette data askingPrice est **optionnelle**, dépendant de la nature du contrat. Le BACS doit savoir si il doit ou non recevoir cette information.

• L'opérateur envoie la confirmation de la requête (selon délai de mobilisation), identifiée par son requestID, au BACS si celui ci a accepté la demande initiale.

func confirmFlexRequest(requestID: String, bacs: BACSID, asset: AssetID, flexProduct :

Protocole : Timeout : 1s. Pour accuser réception.
L'absence d'envoi de **confirmFlexRequest** dans un délai de 1 heure après acceptation du BACS (à askFlex) vaut annulation de la requête initiale.

• Le BACS confirme, dans un délai défini dans le contrat (15' type), son acceptation de la demande et s'engage à l'exécuter.

func acceptFlexRequest(requestID: String, bacs: BACSID, asset: AssetID, flexProduct :

Protocole : Timeout pour réponse opérateur : 5s. En cas de non réponse (ack) de l'opérateur, l'acceptation peut être réémise 1 fois. Au delà, le BACS log l'absence de réponse.
Le seul paramètre requestID est suffisant ; les autres paramètres ne sont qu'une confirmation.

• Annulation de la confirmation de requête : absent en V1

func cancelConfirmation(requestID: String) → Ack

2.2.2.3. API Niveau 3

Au niveau 3, le BACS a la capacité d'estimer sa consommation et donc sa flexibilité à venir pour faire des propositions à l'opérateur et s'engage contractuellement avec l'opérateur de flexibilité.

l'API fournit une information “temps réel” (15') à l'opérateur, sur le potentiel de flexibilité prévisionnelle des assets.

• Le BACS envoie une proposition de flexibilité potentielle à l'opérateur de flex; plusieurs propositions peuvent être envoyées. Cette proposition ne contient pas de proposition de prix. La Reponse est soit acceptée, refusée, à modifier.

func sendPotentielFlex(proposalID: String, bacs: BACSID, asset: AssetID, flexProduct :

Protocole : Timeout pour réponse opérateur : 1h. En cas de non réponse de l'opérateur, la proposition n'est pas réémise.

• L'opérateur envoie au BACS une demande de flexibilité, basée sur la proposition du BACS, avec proposition de prix éventuelle. Le BACS l'accepte ou la refuse ou demande de renégocier dans sa Reponse. Cette API est identique à celle définie au niveau 2. Elle est mentionnée ici pour mémoire.

func askFlexWithPrice(bacs: BACSID, asset: AssetID, flexProduct : FlexProductType, puissance:

• En cas de réponse d'acceptation par le BACS, l'opérateur valide, dans un délai défini dans le contrat, la prise en compte de la proposition et attend un acquittement qui confirme l'acceptation pour exécution.

func validateFlexProposal(proposalID: String, bacs: BACSID, asset: AssetID, flexProduct :

Protocole : Timeout : 1s. Pour accuser réception.
L'absence d'envoi de **validateFlexProposal** dans un délai de 1 heure (?) vaut annulation de la requête initiale. Sera suivi d'un confirmFlexRequest.

2. Validation des API

Pour obtenir le label FlexReady, avec niveau, le fournisseur du BACS devra avoir vérifié qu'il respecte les interfaces définies.
Dans un premier temps, ceci sera fait par auto déclaration[2], après exécution des scenarios de test sur l'émulateur.
Pour cela, FlexReady met à disposition des acteurs :

- un émulateur de serveur d'opérateur de flexibilité
- un émulateur de BACS avec les différents niveaux
- la définition de la séquence de test à effectuer (incluant des cas de données erronées).

L'émulateur fournit un rapport de test en fin de séquence, à joindre à l'auto déclaration

3. Scenarios de test

L'échange des Identifiants (eg., BACS ID) est prédéfini au moment du contrat et ne fait pas partie de l'API. Ils sont entrés « en dur » dans l'émulateur.

Pour chaque API :
- Echange d'authentification, selon le protocole retenu (Oauth 2., à confirmer)

- Si l'identifiant (e.g., BACSID) n'est pas reconnu, le BACS ignore la requête et répond « Erreur ». Après une répétition (5?) d'identifiant non valide, le BACS alerte d'une possible tentative d'intrusion ou de déni de service.

- Envoi d'une requête valide
- Réponse : une fois en acceptation, une fois en refus

2 Voir document SBA Connect un exemple d'auto déclaration: Cadre de référence R2S Connect - Smart Buildings Alliance

- Test de requêtes non valides (sur les dates, sur les AssetID, sur le volume…), Les scenarios de test se présentent selon les diagrammes ci-dessous :

2.3. Prérequis de bonnes pratiques cybersécurité

A venir

2.4. Connexion aux compteurs du gestionnaire de réseau de distribution

2.4.1. Objet de la connexion système de pilotage énergétique - Compteur.

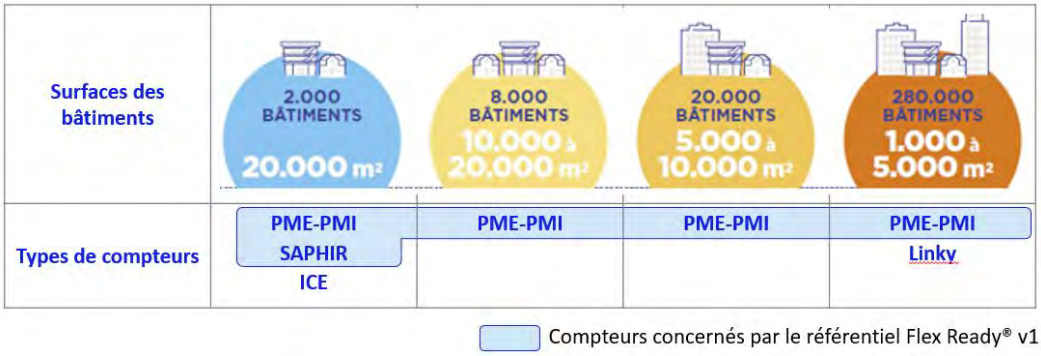
Ce chapitre définit la partie **« connexion au compteur »** entre un système de pilotage énergétique et le compteur du gestionnaire de réseau public de distribution (GRD) telle que prévue dans le référentiel Flex Ready®. C'est une condition optionnelle à l'obtention du droit d'utilisation de la marque Flex Ready®, en complément des prérequis définis par ailleurs. La connexion au compteur offre en effet des informations supplémentaires à la maille globale du bâtiment utiles au pilotage des usages dans le cadre de services de flexibilités pour le système électrique ou les réseaux.

Pour la première version v1 du référentiel Flex Ready®, la connexion au compteur est permise par l'interface de données temps réel (Télé-information Client – TIC) accessible sur les compteurs du GRD. Il s'agit d'un des constituants du référentiel Flex Ready®, celui-ci étant construit pour être mis en œuvre de façon progressive.

Pour des versions ultérieures du référentiel Flex Ready®, d'autres modalités de connexion aux compteurs du GRD seront envisageables, en fonction des évolutions à venir sur les chaines de comptage du Marché d'Affaires (sites raccordés en HTA ou basse tension avec puissance électrique supérieure à 36 kVA). Elles feront l'objet d'une actualisation du référentiel Flex Ready®.

2.4.2. Les modalités de connexion au compteur GRD.

2.4.2.1. Les différents types de compteurs GRD



Les bâtiments tertiaires sont en grande majorité en BT > 36 kVA. Pour ce niveau de puissance, le compteur GRD est le PME-PMI.

Les sites tertiaires HTA ont majoritairement un compteur SAPHIR (le compteur ICE, ancien compteur HTA, est en extinction et certains *client HTA ont comptage BT PME-PMI au secondaire*[2]). Les sites avec un compteur Linky (≤ 36 kVA) ne sont a priori pas concernés par le décret BACS.

2.4.2.2. La connexion aux compteurs PME-PMI et Saphir

Pour se connecter à la TIC des compteurs PME-PMI et SAPHIR il faut se référer à la **note technique Enedis-NOI-CPT_02E**, disponible en ligne, qui décrit les caractéristiques physiques des TIC des compteurs et les données transmises.

La TIC PME-PMI est disponible sur un port RJ45. L'ensemble des caractéristiques physiques des signaux échangés est conforme à la norme RS232. La TIC PME-PMI est non modulée.

La TIC SAPHIR suit la norme Euridis. Elle est modulée en amplitude avec un signal sur porteuse et le port physique est un bornier à vis 2 fils.

[2] *Voir le paragraphe « 5.1.2. Point de Livraison HTA avec comptage en BT » de la note Enedis-NMO-CPT_002E, disponible en ligne.*

2.4.3. Les critères de connexion au compteur GRD du système de pilotage énergétique.

Pour bénéficier de la marque Flex Ready® sur la partie connexion au compteur du GRD, un système de pilotage énergétique doit respecter la totalité des critères suivants :

- *Critère 1 : Le* système de pilotage énergétique *est en mesure de se connecter physiquement à la TIC du compteur PME-PMI et/ou SAPHIR, de manière directe ou indirecte (par exemple via une passerelle de communication).*

- *Critère 2 : Le* système de pilotage énergétique *récupère a minima les données suivantes émises par la TIC du compteur PME-PMI et/ou SAPHIR :*

Compteur	Données TIC	Cas d'usage associé
TIC PME-PMI	- **[DATE] :** Horadate courante du compteur (utilisée pour la facturation) sous le format « JJ/MM/AA HH:MM:SS ».	Synchronisation avec l'horloge du compteur
	- **[PREAVIS]** : émis que lorsque la puissance atteinte mesurée 1 mn est supérieure au seuil de 90% de la puissance souscrite de la période tarifaire en cours. - **[PS]** : puissance souscrite de la période tarifaire en cours (en kVA en BT > 36 kVA et kW en HTA), qui peut changer selon la période tarifaire. - **[EAPP_s]** : énergie apparente soutirée depuis le dernier top Td (temps d'intégration utilisé dans le calcul des dépassements, usuellement 5 ou 10mn et synchronisé avec les minutes rondes de l'horloge du compteur)**.** Pour les clients BT Sup 36, cette donnée peut être utilisée pour s'assurer chaque minute du non dépassement de la puissance souscrite lors des Td dernières minutes.[3] - **[EA_s]** : énergie active soutirée depuis le dernier top Td (temps d'intégration utilisé dans le calcul des dépassements, usuellement 5 ou 10mn et synchronisé avec les minutes rondes de l'horloge du compteur). *Concerne uniquement les cas particuliers d'un client HTA avec comptage BT PME-PMI au secondaire.* Cette donnée peut être utilisée pour s'assurer chaque Td minute du non	Asservissement à la puissance souscrite contractuelle

[3] *Pour plus de détails sur le calcul des dépassements en BT > 36 kVA, voir le paragraphe « 2.2.5. Précisions sur le calcul de la puissance atteinte et des dépassements de la puissance contractuelle de référence » de la note Enedis-NOI-CPT_36E, disponible en ligne.*

	dépassement de la puissance souscrite lors des Td dernières minutes[4].	
	- **[PTCOUR1] :** Période tarifaire courante (chaîne associée de 3 caractères alphanumériques) du calendrier distributeur. - **[PTCOUR2] :** Période tarifaire courante (chaîne associée de 3 caractères alphanumériques) du calendrier fournisseur.	Pilotage tarifaire
	- **[EAP_s] :** Energie active soutirée de la période tarifaire en cours du calendrier distributeur. - **[EaP_s2] :** Energie active soutirée de la période tarifaire en cours du calendrier fournisseur.	Suivi de la consommation
TIC SAPHIR	- **[DATE] :** Horadate courante du compteur (utilisée pour la facturation) sous le format « JJ/MM/AA HH/MM/SS ».	Synchronisation avec l'horloge du compteur
	- **[PREAVIS] :** émis que lorsque la puissance active soutirée est supérieure au seuil de [KDC]= XXX% de la puissance souscrite de la période tarifaire en cours et n'est plus émis lorsque que la puissance active soutirée est inférieure à [KDCD]=XXX%. - **[KDC] :** seuil d'avertissement de dépassement de puissance (en %). - **[KDCD] :** seuil de fin d'avertissement de dépassement de puissance souscrite (en %). - **[PSpx] :** puissance souscrite de la période x en cours du calendrier distributeur (en kW). - **[PTCOURD] :** Période tarifaire courante (chaîne associée de 3 caractères alphanumériques) du calendrier distributeur. - **[LIB_pxD] :** Libellé de la période tarifaire n° x (x=1 à 8) de la grille D. - **[EAS] :** Energie active en soutirage depuis le dernier top Td, qui est le temps d'intégration utilisé dans le calcul des dépassements (usuellement5 ou 10 mn et synchronisé avec les minutes rondes de l'horloge du compteur). Cette donnée peut être utilisée pour s'assurer chaque Td minute du non dépassement de la puissance souscrite lors des Td dernières minutes[5].	Asservissement à la puissance souscrite contractuelle

[4] *Pour plus de détails sur le calcul des dépassements en HTA, voir le paragraphe « 2.2.4. Précisions sur le calcul de la puissance atteinte et des dépassements de la puissance contractuelle de référence en soutirage » de la note Enedis-NOI-CPT_56E, disponible en ligne.*

[5] *Pour plus de détails sur le calcul des dépassements en HTA, voir le paragraphe « 2.2.4. Précisions sur le calcul de la puissance atteinte et des dépassements de la puissance contractuelle de référence en soutirage » de la note Enedis-NOI-CPT_56E, disponible en ligne.*

	- **[PTCOURF] :** Période tarifaire courante (chaîne associée de 3 caractères alphanumériques) du calendrier fournisseur. - **[PTCOURD] :** Période tarifaire courante (chaîne associée de 3 caractères alphanumériques) du calendrier distributeur.	Pilotage tarifaire
	- **[EApxSD] :** Index d'énergie active soutirée de la période tarifaire n° x (x=1 à 8) de la grille distributeur. - **[EApxSF] :** Index d'énergie active soutirée de la période tarifaire n° x (x=1 à 8) et de grille fournisseur.	Suivi de la consommation

ANNEXE 1 – Annexe technique de l'API Flex Ready® et fonctionnement de l'émulateur *(English only)*

This document includes a detailed description of the dataset, endpoint specifications (with the corresponding function names in the titles), and a summary of the core endpoint functions. This annex serves as a reference for understanding the data model, API integration, and overall architecture of the emulator.

Dataset Overview

The emulator is seeded with data that mimics a BACS environment. The data is structured as follows:

1. BACS Data

The dataset contains two BACS records:

- **BACS 1:**
  - **ID:** `f47ac10b-58cc-4372-a567-0e02b2c3d479`
  - **Name:** "BACS Whole Building Only"
  - **Description:** A BACS configured for a whole building that has a single associated asset.
- **BACS 2:**
  - **ID:** `a987fbc9-4bed-4078-9f07-9141ba07c9f3`
  - **Name:** "BACS Whole Building with Heating System"
  - **Description:** A BACS configured for a whole building that has multiple associated assets.

2. Assets Data

Three asset records are provided:

- **Asset 1 (BACS 1):**
  - **Asset ID:** `WholeBuilding`
  - **BACS ID:** `f47ac10b-58cc-4372-a567-0e02b2c3d479`
  - **Flex Product:** `"WID"`
  - **Potential (Potentiel):**
    - **Power:** `{ "unit": "W", "multiplier": "k", "value": 100.0 }`
    - **Mobilization Delay:** `1` minute
    - **Max Duration:** `2` minutes
    - **Activation Per Day:** `1`
    - **Time Period:** A single period from `"2025-04-02T12:00:00"` to `"2025-04-02T14:00:00"`
- **Asset 2 (BACS 2):**
  - **Asset ID:** `WholeBuilding`

- **BACS ID:** a987fbc9-4bed-4078-9f07-9141ba07c9f3
- **Flex Product:** "WID"
- **Potential:**
  - **Power:** { "unit": "W", "multiplier": "k", "value": 80.0 }
  - **Mobilization Delay:** 1 minute
  - **Max Duration:** 2 minutes
  - **Activation Per Day:** 1
  - **Time Period:** A single period from "2025-04-02T10:00:00" to "2025-04-02T12:00:00"

- **Asset 3 (BACS 2):**

  - **Asset ID:** IRVE
  - **BACS ID:** a987fbc9-4bed-4078-9f07-9141ba07c9f3
  - **Flex Product:** "WID"
  - **Potential:**

- **Power:** { "unit": "W", "multiplier": "k", "value": 50.0 }
- **Mobilization Delay:** 1 minute
- **Max Duration:** 2 minutes
- **Activation Per Day:** 1
- **Time Period:** A single period from "2025-04-02T14:00:00" to "2025-04-02T16:00:00"

3. Historical Consumption Data

Historical consumption is recorded for assets:

- **Record 1:**
  - **BACS ID:** f47ac10b-58cc-4372-a567-0e02b2c3d479
  - **Asset ID:** WholeBuilding
  - **Date:** "2025-04-01"
  - **Hour:** "12:00:00"
  - **Consumption:** { "unit": "W", "multiplier": "k", "value": 50.0 }
- **Record 2:**
  - **BACS ID:** f47ac10b-58cc-4372-a567-0e02b2c3d479
  - **Asset ID:** WholeBuilding
  - **Date:** "2025-04-01"
  - **Hour:** "12:15:00"
  - **Consumption:** { "unit": "W", "multiplier": "k", "value": 45.5 }

4. Flex Requests Consumption Data

This dataset contains consumption reports for flex requests:

- **Record 1:**
  - **BACS ID:** f47ac10b-58cc-4372-a567-0e02b2c3d479
  - **Asset:** WholeBuilding
  - **Reported Entries:**
    - **Entry 1:**
      - **Power:** { "unit": "W", "multiplier": "k", "value": 80.0 }
      - **Start:** "2025-04-02T10:00:00"
      - **End:** "2025-04-02T10:15:00"
    - **Entry 2:**
      - **Power:** { "unit": "W", "multiplier": "k", "value": 75.0 }
      - **Start:** "2025-04-03T11:00:00"
      - **End:** "2025-04-03T12:15:00"

Record 2:

- **BACS ID:** a987fbc9-4bed-4078-9f07-9141ba07c9f3
- **Asset:** IRVE

- **Reported Entry:**
  - **Power:** `{ "unit": "W", "multiplier": "k", "value": 50.0 }`
  - **Start:** `"2025-04-05T09:00:00"`
  - **End:** `"2025-04-05T09:15:00"`
- **Record 3:**
  - **BACS ID:** `a987fbc9-4bed-4078-9f07-9141ba07c9f3`
  - **Asset:** `WholeBuilding`
  - **Reported Entry:**
    - **Power:** `{ "unit": "W", "multiplier": "k", "value": 50.0 }`
    - **Start:** `"2025-04-05T09:00:00"`
    - **End:** `"2025-04-05T09:15:00"`

Endpoints Descriptions

Each endpoint in the Emulator is defined by its HTTP method and URL structure. In every request you must include standard headers such as:

- **Accept:** e.g. `application/json` (or `application/pdf` for reports)
- **Content-Type:** e.g. `application/json` (for POST endpoints)
- **Authorization:** `Bearer <your_jwt_token>`

For endpoints under the Operator interface (those whose URLs contain `/operator/`), you can also include additional headers that start with the prefix `EMULATOR_` (for example, `EMULATOR_Authorization: Bearer <your_jwt_token>`) that are extracted by the proxy and forwarded to the external service.

BACS Endpoints

- **Retrieve Assets and Flexibility Potential (getBACSAssets) Method:**

GET
URL: `/{session_id}/bacs/{bacs_id}/assets`
Request Body: *None*
Return: A JSON list of FlexibilityAsset objects.

- **Create Flex Request (askFlex and askFlexWithPrice if askingPrice is given) Method:**

POST
URL: `/{session_id}/bacs/{bacs_id}/assets/{asset_id}/flexibilities/request`
Request Body:
Note: askingPrice is only required for level 2 of flexibility

```
{
"requestID": "<request_id>", "flexProduct":
"<flex_product_code>", "power": [
{
"power": { "unit": "<unit>", "multiplier": "<multiplier>", "value": <value> }, "start":
"<YYYY-MM-DDTHH:MM:SS>",
"end": "<YYYY-MM-DDTHH:MM:SS>"
      }
    ],
```

Return:

```
{ "message": "Flex request created successfully" }
```

- Confirm Flex Request (confirmFlexRequest) Method:

POST
URL: `/{session_id}/bacs/{bacs_id}/assets/{asset_id}/flexibilities/request/{request_id}/confirm`
Request Body:

```
{
"flexProduct": "<flex_product_code>", "power":
[
{
"power": { "unit": "<unit>", "multiplier": "<multiplier>", "value": <value> }, "start":
"<YYYY-MM-DDTHH:MM:SS>",
"end": "<YYYY-MM-DDTHH:MM:SS>"
      }
```

Return:

```
{ "message": "Flex request confirmed successfully" }
```

- Realize Flex Request (realiseFlexRequest) Method:

GET
URL: /{session_id}/bacs/{bacs_id}/assets/{asset_id}/flexibilities/request/{request_id}/consumptions
Request Body: None
Return: A JSON list of RealizedConsumptionResponse objects.

- Retrieve Detailed Asset Consumption (getBACSAssetsHisto) Method:

GET
URL: /{session_id}/bacs/{bacs_id}/assets/{asset_id}/consumptions
Request Body: None
Return: A JSON list of historical consumption records.

- Retrieve Aggregated Asset Consumption (getBACSAssetsConsos)

Method: GET
URL: /{session_id}/bacs/{bacs_id}/assets/consumptions
Request Body: None
Return: A JSON object summarizing consumption data.

Operator Endpoints

- Retrieve Assets via Operator Interface Method:

GET
URL: /{session_id}/operator/bacs/{bacs_id}/assets
Request Body: None
Return: A JSON object containing a list of assets.\

- Create Flex Request via Operator Interface

Method: POST
URL: /{session_id}/operator/bacs/{bacs_id}/assets/{asset_id}/flexibilities/request
Request Body:

```
{
"requestID": "<request_id>", "flexProduct":
"<flex_product_code>", "power": [
{
"power": { "unit": "<unit>", "multiplier": "<multiplier>", "value": <value> }, "start":
"<YYYY-MM-DDTHH:MM:SS>",
"end": "<YYYY-MM-DDTHH:MM:SS>"
      }
    ],
```

Return:

```
{ "data": { ... }, "test_status": "NOMINAL TEST AND ROBUSTNESS TESTS PASSED" }
```

- Confirm Flex Request via Operator Interface Method:

POST

URL: `/{session_id}/operator/bacs/{bacs_id}/assets/{asset_id}/flexibilities/request/{request_id}/confirm`

Request Body:

```
{
"flexProduct": "<flex_product_code>", "power":
[
{
"power": { "unit": "<unit>", "multiplier": "<multiplier>", "value": <value> }, "start":
"<YYYY-MM-DDTHH:MM:SS>",
"end": "<YYYY-MM-DDTHH:MM:SS>"
      }
```

Return:

```
{ "message": "Flex request confirmed successfully" }
```

- Realize Flex Request via Operator Interface

Method: GET
URL:
`/{session_id}/operator/bacs/{bacs_id}/assets/{asset_id}/flexibilities/request/{request_id}/consumptions`
Request Body: None
Return: A JSON list of realized consumption data.\

- Retrieve Detailed Asset Consumption via Operator Interface Method:

GET
URL: `/{session_id}/operator/bacs/{bacs_id}/assets/{asset_id}/consumptions`
Request Body: None
Return: A JSON list of historical consumption records.\

- Retrieve Aggregated Asset Consumption via Operator Interface Method:

GET
URL: `/{session_id}/operator/bacs/{bacs_id}/assets/consumptions`
Request Body: None
Return: A JSON object summarizing consumption data.\

Test Report Endpoint

- Download Test Report Method:

POST
URL: `/{session_id}/report`
Request Body: None
Return: A PDF file of the test session report.

Test Session Validation Guide

This guide provides step-by-step instructions to validate a test session for the Emulator for BACS Flexibility Management. Two scenarios are covered:

- **Operator Mode:** Using the BACS interface endpoints: you are an operator that wants to test the emulator's BACS (without additional `EMULATOR_*` headers). To validate a test session as an operator, you need to execute `askflex`, `confirmflexrequest`, and

`realiseflexrequest` with the level of flexibility 2.

- **BACS Mode:** Using the Operator interface endpoints. You are a BACS constructor that wants to be tested against the emulator's operator: to validate the test session you need to execute `askflex`, `confirmflexrequest`, and `realiseflexrequest`. If you choose the level of flexibility 2, you also need to execute `getbacsassetsconsos` and `getbacsassetshisto`.

All requests to protected endpoints require a valid JWT token in the `Authorization: Bearer <token>` header. For Operator endpoints (i.e., URLs containing `/operator/`), you can also include any extra headers prefixed with `EMULATOR_` (e.g., `EMULATOR_Authorization: Bearer <token>`) that will be extracted and forwarded.

Replace the placeholders `{session_id}`, `{bacs_id}`, `{asset_id}`, `{requestID}`, and `<your_jwt_token>` with the actual values obtained during execution.

Common Steps

1. Authentication (Login)

Obtain a JWT token by providing your credentials:

```
curl -X POST "https://flexready-emulator.heka.ai/api/v1/auth/login" \
-H "Accept: application/json" \
-H "Content-Type: application/x-www-form-urlencoded" \
```

A successful response looks like:

```
{
"access_token": "<your_jwt_token>", "token_type":
"bearer"
```

If you don't have an account to access the emulator. Please complete this form : https://forms.office.com/e/nanU0zM0Td?origin=lprLink

2. Create Test Session

Create a test session by specifying the flexibility level(1 or 2), user type, and your external URL.

- For BACS Mode (user type: BACS, you are a BACS constructor that wants to be tested against the emulator's operator):

```
curl -X POST "https://flexready-emulator.heka.ai/api/v1/sessions?
level_of_flexibility=2&user=BACS&url=https://your-example-url.com" \
-H "Accept: application/json" \
```

For Operator Mode (user type: Operator, you are an operator that wants to test the emulator's BACS):

```
curl -X POST "https://flexready-emulator.heka.ai/api/v1/sessions?
level_of_flexibility=2&user=Operator&url=https://your-example-url.com" \
-H "Accept: application/json" \
```

The response will include a session identifier (`session_id`) to be used for requests.

Part A – Validation as Operator (Using the BACS Interface)

3. Retrieve Assets and Flexibility Potential

Generic Example:

```
curl -X GET "https://flexready-emulator.heka.ai/api/v1/{session_id}/bacs/{bacs_id}/assets" \
-H "Accept: application/json" \
```

Specific Example (using BACS 1, asset: WholeBuilding):

```
curl -X GET "https://flexready-emulator.heka.ai/api/v1/{session_id}/bacs/f47ac10b-58cc-4372-a567-0e02b2c3d479/assets" \
-H "Accept: application/json" \
```

4. Create Flex Request

Note : `askingPrice` field is mandatory only for second level of flexibility.

Generic Example:

```
curl -X POST "https://flexready-emulator.heka.ai/api/v1/{session_id}/bacs/{bacs_id}/assets/{asset_id}/flexibilities/request" \
-H "Accept: application/json" \
-H "Content-Type: application/json" \
-H "Authorization: Bearer <your_jwt_token>" \
-d '{
"requestID": "{request_id}", "flexProduct": "<flex_product_code>", "power": [
{
"power": { "unit": "<unit>", "multiplier": "<multiplier>", "value": <value> }, "start": "<YYYY-MM-DDTHH:MM:SS>",
"end": "<YYYY-MM-DDTHH:MM:SS>"
```

Specific Example (using BACS 1, asset: WholeBuilding, request ID: req-001):

```
curl -X POST "https://flexready-emulator.heka.ai/api/v1/{session_id}/bacs/f47ac10b-58cc-4372-a567-0e02b2c3d479/assets/WholeBuilding/flexibilities/request" \
-H "Accept: application/json" \
-H "Content-Type: application/json" \
-H "Authorization: Bearer <your_jwt_token>" \
-d '{
"requestID": "req-001", "flexProduct": "WID", "power": [
{
"power": { "unit": "W", "multiplier": "k", "value": 80.0 }, "start": "2025-04-02T10:00:00",
"end": "2025-04-02T12:00:00"
```

5. Confirm Flex Request

Generic Example:

```
curl -X POST "https://flexready-
emulator.heka.ai/api/v1/{session_id}/bacs/{bacs_id}/assets/{asset_id}/flexibilities/request/{request_id}/
confirm" \
-H "Accept: application/json" \
-H "Content-Type: application/json" \
```

```
"flexProduct": "<flex_product_code>", "power": [

{

"power": { "unit": "<unit>", "multiplier": "<multiplier>", "value": <value> }, "start": "<YYYY-MM-
DDTHH:MM:SS>",

"end": "<YYYY-MM-DDTHH:MM:SS>"

            }

          ],

"askingPrice": { "unit": "<unit>", "multiplier": "<multiplier>", "value": <value> }
```

Specific Example:

```
curl -X POST "https://flexready-emulator.heka.ai/api/v1/{session_id}/bacs/f47ac10b-58cc-4372-a567-
0e02b2c3d479/assets/WholeBuilding/flexibilities/request/req-001/confirm" \

-H "Accept: application/json" \

-H "Content-Type: application/json" \

-H "Authorization: Bearer <your_jwt_token>" \

-d '{

"flexProduct": "WID", "power": [

{

"power": { "unit": "W", "multiplier": "k", "value": 80.0 }, "start": "2025-04-
02T10:00:00",

"end": "2025-04-02T12:00:00"
```

6. Realize Flex Request

Generic Example:

```
curl -X GET "https://flexready-
emulator.heka.ai/api/v1/{session_id}/bacs/{bacs_id}/assets/{asset_id}/flexibilities/request/{request_id}/
consumptions" \

-H "Accept: application/json" \
```

Specific Example:

```
curl -X GET "https://flexready-emulator.heka.ai/api/v1/{session_id}/bacs/f47ac10b-58cc-4372-a567-
0e02b2c3d479/assets/WholeBuilding/flexibilities/request/req-001/consumptions" \

-H "Accept: application/json" \
```

7. Retrieve Asset Consumption Data

a) Detailed Consumption

Generic Example:

```
curl -X GET "https://flexready-
emulator.heka.ai/api/v1/{session_id}/bacs/{bacs_id}/assets/{asset_id}/consumptions" \
-H "Accept: application/json" \
```

Specific Example:

```
curl -X GET "https://flexready-emulator.heka.ai/api/v1/{session_id}/bacs/f47ac10b-58cc-4372-a567-0e02b2c3d479/assets/WholeBuilding/consumptions" \
-H "Accept: application/json" \
```

b) Retrieve consumption of all assets of a given BACS

Generic Example:

```
curl -X GET "https://flexready-emulator.heka.ai/api/v1/{session_id}/bacs/{bacs_id}/assets/consumptions" \
-H "Accept: application/json" \
```

Specific Example:

```
curl -X GET "https://flexready-emulator.heka.ai/api/v1/{session_id}/bacs/a987fbc9-4bed-4078-9f07-9141ba07c9f3/assets/consumptions" \
-H "Accept: application/json" \
```

8. Download Test Report

Generic Example:

```
curl -X POST "https://flexready-emulator.heka.ai/api/v1/sessions/{session_id}/report" \
-H "Accept: application/pdf" \
-H "Authorization: Bearer <your_jwt_token>" \
```

Part B – Validation as BACS (Using the Operator Interface)

Note: For Operator endpoints, you can include extra headers with the prefix `EMULATOR_` (the extra header may be any header that starts with `EMULATOR_`, e.g., `EMULATOR_Authorization`, `EMULATOR_X-API-Key`, etc.) that are forwarded to the external service.

3. Retrieve Assets via Operator Interface

Generic Example:

```
curl -X GET "https://flexready-emulator.heka.ai/api/v1/{session_id}/operator/bacs/{bacs_id}/assets" \
-H "Accept: application/json" \
-H "Authorization: Bearer <your_jwt_token>" \
```

(The extra header can be any header starting with `EMULATOR_`.)

4. Create Flex Request (Operator)

Generic Example:

```
curl -X POST "https://flexready-
emulator.heka.ai/api/v1/{session_id}/operator/bacs/{bacs_id}/assets/{asset_id}/flexibilities/request" \

-H "Accept: application/json" \
```

```
-H "Content-Type: application/json" \
-H "Authorization: Bearer <your_jwt_token>" \
-H "EMULATOR_<any_header>: <any_value>" \
-d '{
"requestID": "{request_id}", "flexProduct":
"<flex_product_code>", "power": [
{
"power": { "unit": "<unit>", "multiplier": "<multiplier>", "value": <value> }, "start": "<YYYY-MM-
DDTHH:MM:SS>",
"end": "<YYYY-MM-DDTHH:MM:SS>"
            }
```

5. Confirm Flex Request (Operator)

Generic Example:

```
curl -X POST "https://flexready-
emulator.heka.ai/api/v1/{session_id}/operator/bacs/{bacs_id}/assets/{asset_id}/flexibilities/request/{req
uest_id}/confirm" \
-H "Accept: application/json" \
-H "Content-Type: application/json" \
-H "Authorization: Bearer <your_jwt_token>" \
-H "EMULATOR_<any_header>: <any_value>" \
-d '{
"flexProduct": "<flex_product_code>", "power": [
{
"power": { "unit": "<unit>", "multiplier": "<multiplier>", "value": <value> }, "start": "<YYYY-MM-
DDTHH:MM:SS>",
"end": "<YYYY-MM-DDTHH:MM:SS>"
```

6. Realize Flex Request (Operator)

Generic Example:

```
curl -X GET "https://flexready-
emulator.heka.ai/api/v1/{session_id}/operator/bacs/{bacs_id}/assets/{asset_id}/flexibilities/request/{req
uest_id}/consumptions" \
-H "Accept: application/json" \
-H "Authorization: Bearer <your_jwt_token>" \
```

7. Retrieve Asset Consumption Data via Operator Interface

a) Detailed Consumption

Generic Example:

```
curl -X GET "https://flexready-
emulator.heka.ai/api/v1/{session_id}/operator/bacs/{bacs_id}/assets/{asset_id}/consumptions" \
-H "Accept: application/json" \
```

```
-H "Authorization: Bearer <your_jwt_token>" \
-H "EMULATOR_<any_header>: <any_value>"
```

b) Retrieve consumption of all assets of a given BACS

Generic Example:

```
curl -X GET "https://flexready-
emulator.heka.ai/api/v1/{session_id}/operator/bacs/{bacs_id}/assets/consumptions" \
-H "Accept: application/json" \
-H "Authorization: Bearer <your_jwt_token>" \
```

8. Download Test Report

Generic Example:

```
curl -X POST "https://flexready-emulator.heka.ai/api/v1/sessions/{session_id}/report" \
-H "Accept: application/pdf" \
-H "Authorization: Bearer <your_jwt_token>" \
```

ANNEXE 2 - Process général d'échanges (issu IEC 62746-4).

LE CHAMP DE L'API FLEX READY® EST CADRE EN ROUGE.

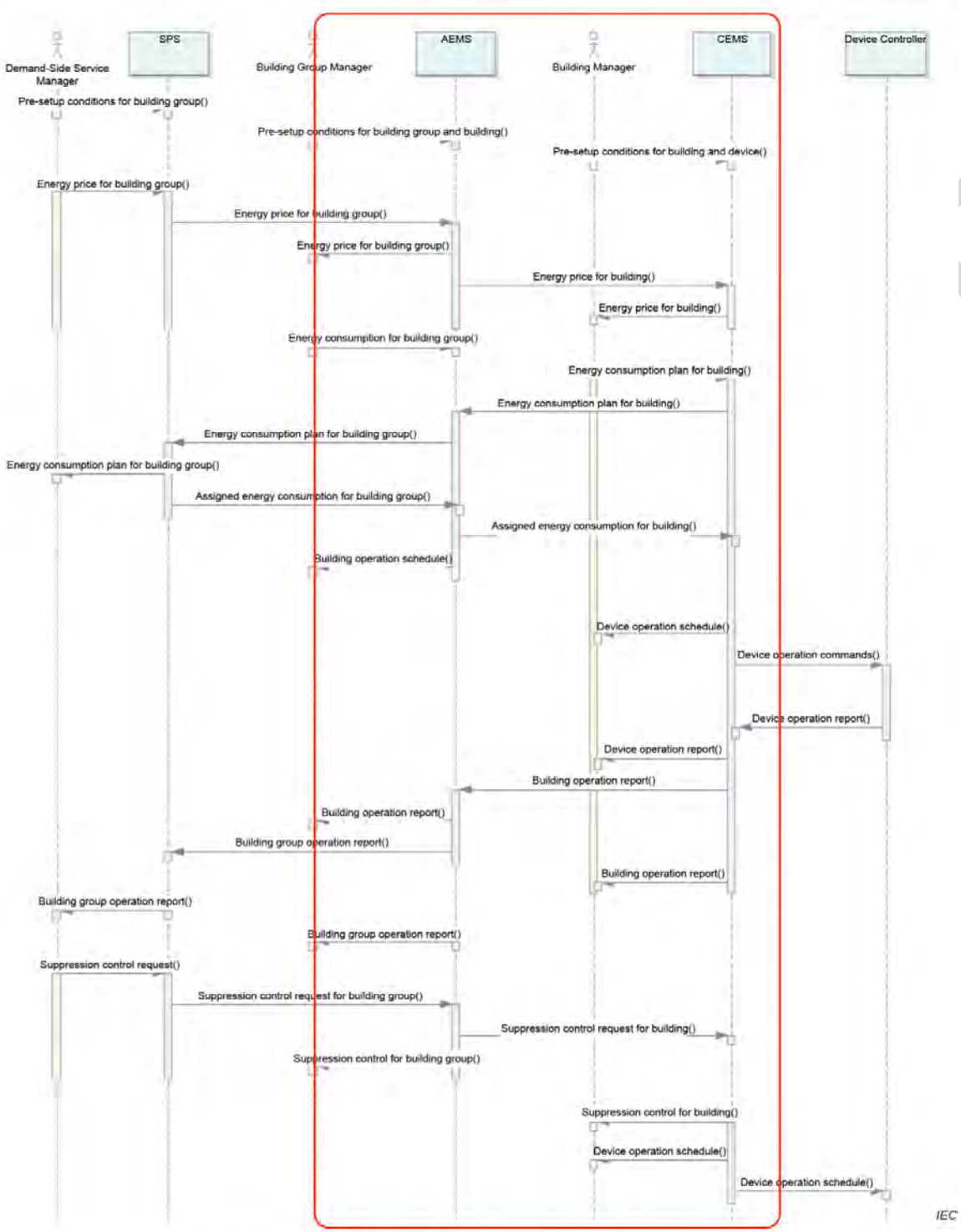


Figure A.2 – The whole view of this use case

ANNEXE **3 - Définition des acronymes**

Acronyme	Signification
ACER	Agency for the Cooperation of Energy Regulators
API	Application Programming Interface
BACS	Building Automation and Control Systems
BT	Basse Tension
CO2	Dioxyde de Carbone
DP	Point de Livraison
EA	Énergie Active
EAPP	Énergie Apparente
EMS	Energy Management System
ENR	Énergies Renouvelables
ENTSOe	European Network of Transmission System Operators for Electricity
EU DSO	European Distribution System Operators
FE	Fournisseur d'Électricité
FM/PM	Facility Management/Property Management
GMT	Greenwich Mean Time
GR	Gestionnaires de Réseaux
GRD	Gestionnaire de Réseau de Distribution
HTA	Haute Tension A
ID	Identifiant
IHM	Interface Homme-Machine
INPI	Institut National de la Propriété Industrielle
JSON	JavaScript Object Notation
KDC	Seuil d'Avertissement de Dépassement de Puissance
KDCD	Seuil de Fin d'Avertissement de Dépassement de Puissance
kVA	Kilovoltampère
kW	Kilowatt
MT	Moyen Terme
OE	Opérateurs d'Effacement
PME-PMI	Petites et Moyennes Entreprises - Petites et Moyennes Industries
REX	Retour d'Expérience
RTE	Réseau de Transport d'Électricité
SIA	Système d'Information et d'Aide
TIC	Télé-information Client

TSG	Think Smartgrids
UUID	Universally Unique Identifier
WDA	Wholesale Day Ahead Market
WID	Wholesale IntraDay Market