Table of Contents

WP-7328: NetApp Conversational AI Using NVIDIA Jarvis ................................................................. 1
  Solution Overview ........................................................................................................................... 1
  Solution Technology ......................................................................................................................... 3
  Overview ........................................................................................................................................ 5
  Conclusion ....................................................................................................................................... 19
  Acknowledgments ............................................................................................................................ 20
  Where to Find Additional Information ............................................................................................. 20
The NVIDIA DGX family of systems is made up of the world’s first integrated artificial intelligence (AI)-based systems that are purpose-built for enterprise AI. NetApp AFF storage systems deliver extreme performance and industry-leading hybrid cloud data-management capabilities. NetApp and NVIDIA have partnered to create the NetApp ONTAP AI reference architecture, a turnkey solution for AI and machine learning (ML) workloads that provides enterprise-class performance, reliability, and support.

This white paper gives directional guidance to customers building conversational AI systems in support of different use cases in various industry verticals. It includes information about the deployment of the system using NVIDIA Jarvis. The tests were performed using an NVIDIA DGX Station and a NetApp AFF A220 storage system.

The target audience for the solution includes the following groups:

- Enterprise architects who design solutions for the development of AI models and software for conversational AI use cases such as a virtual retail assistant
- Data scientists looking for efficient ways to achieve language modeling development goals
- Data engineers in charge of maintaining and processing text data such as customer questions and dialogue transcripts
- Executive and IT decision makers and business leaders interested in transforming the conversational AI experience and achieving the fastest time to market from AI initiatives

Next: Solution Overview

Solution Overview

NetApp ONTAP AI and Cloud Sync

The NetApp ONTAP AI architecture, powered by NVIDIA DGX systems and NetApp cloud-connected storage systems, was developed and verified by NetApp and NVIDIA. This reference architecture gives IT organizations the following advantages:

- Eliminates design complexities
- Enables independent scaling of compute and storage
- Enables customers to start small and scale seamlessly
- Offers a range of storage options for various performance and cost points NetApp ONTAP AI tightly integrates DGX systems and NetApp AFF A220 storage systems with state-of-the-art networking. NetApp ONTAP AI and DGX systems simplify AI deployments by eliminating design complexity and guesswork. Customers can start small and grow their systems in an uninterrupted manner while intelligently managing data from the edge to the core to the cloud and back.

NetApp Cloud Sync enables you to move data easily over various protocols, whether it’s between two NFS shares, two CIFS shares, or one file share and Amazon S3, Amazon Elastic File System (EFS), or Azure Blob storage. Active-active operation means that you can continue to work with both source and target at the same
time, incrementally synchronizing data changes when required. By enabling you to move and incrementally synchronize data between any source and destination system, whether on-premises or cloud-based, Cloud Sync opens up a wide variety of new ways in which you can use data. Migrating data between on-premises systems, cloud on-boarding and cloud migration, or collaboration and data analytics all become easily achievable. The figure below shows available sources and destinations.

In conversational AI systems, developers can leverage Cloud Sync to archive conversation history from the cloud to data centers to enable offline training of natural language processing (NLP) models. By training models to recognize more intents, the conversational AI system will be better equipped to manage more complex questions from end-users.

**NVIDIA Jarvis Multimodal Framework**

![Diagram of available sources and destinations for Cloud Sync](image)

**NVIDIA Jarvis** is an end-to-end framework for building conversational AI services. It includes the following GPU-optimized services:

- Automatic speech recognition (ASR)
- Natural language understanding (NLU)
- Integration with domain-specific fulfillment services
- Text-to-speech (TTS)
- Computer vision (CV)

Jarvis-based services use state-of-the-art deep learning models to address the complex and challenging task of real-time conversational AI. To enable real-time, natural interaction with an end user, the models need to complete computation in under 300 milliseconds. Natural interactions are challenging, requiring multimodal sensory integration. Model pipelines are also complex and require coordination across the above services.

Jarvis is a fully accelerated, application framework for building multimodal conversational AI services that use an end-to-end deep learning pipeline. The Jarvis framework includes pretrained conversational AI models, tools, and optimized end-to-end services for speech, vision, and NLU tasks. In addition to AI services, Jarvis enables you to fuse vision, audio, and other sensor inputs simultaneously to deliver capabilities such as multi-user, multi-context conversations in applications such as virtual assistants, multi-user diarization, and call
NVIDIA NeMo

NVIDIA NeMo is an open-source Python toolkit for building, training, and fine-tuning GPU-accelerated state-of-the-art conversational AI models using easy-to-use application programming interfaces (APIs). NeMo runs mixed precision compute using Tensor Cores in NVIDIA GPUs and can scale up to multiple GPUs easily to deliver the highest training performance possible. NeMo is used to build models for real-time ASR, NLP, and TTS applications such as video call transcriptions, intelligent video assistants, and automated call center support across different industry verticals, including healthcare, finance, retail, and telecommunications.

We used NeMo to train models that recognize complex intents from user questions in archived conversation history. This training extends the capabilities of the retail virtual assistant beyond what Jarvis supports as delivered.

Retail Use Case Summary

Using NVIDIA Jarvis, we built a virtual retail assistant that accepts speech or text input and answers questions regarding weather, points-of-interest, and inventory pricing. The conversational AI system is able to remember conversation flow, for example, ask a follow-up question if the user does not specify location for weather or points-of-interest. The system also recognizes complex entities such as “Thai food” or “laptop memory.” It understands natural language questions like “will it rain next week in Los Angeles?” A demonstration of the retail virtual assistant can be found in Customize States and Flows for Retail Use Case.

Solution Technology

The following figure illustrates the proposed conversational AI system architecture. You can interact with the system with either speech signal or text input. If spoken input is detected, Jarvis AI-as-service (AlaaS) performs ASR to produce text for Dialog Manager. Dialog Manager remembers states of conversation, routes text to corresponding services, and passes commands to Fulfillment Engine. Jarvis NLP Service takes in text, recognizes intents and entities, and outputs those intents and entity slots back to Dialog Manager, which then sends Action to Fulfillment Engine. Fulfillment Engine consists of third-party APIs or SQL databases that answer user queries. After receiving Result from Fulfillment Engine, Dialog Manager routes text to Jarvis TTS AlaaS to produce an audio response for the end-user. We can archive conversation history, annotate sentences with intents and slots for NeMo training such that NLP Service improves as more users interact with the system.
Hardware Requirements

This solution was validated using one DGX Station and one AFF A220 storage system. Jarvis requires either a T4 or V100 GPU to perform deep neural network computations.

The following table lists the hardware components that are required to implement the solution as tested.

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>T4 or V100 GPU</td>
<td>1</td>
</tr>
<tr>
<td>NVIDIA DGX Station</td>
<td>1</td>
</tr>
</tbody>
</table>

Software Requirements

The following table lists the software components that are required to implement the solution as tested.

<table>
<thead>
<tr>
<th>Software</th>
<th>Version or Other Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>NetApp ONTAP data management software</td>
<td>9.6</td>
</tr>
<tr>
<td>Cisco NX-OS switch firmware</td>
<td>7.0(3)I6(1)</td>
</tr>
<tr>
<td>NVIDIA DGX OS</td>
<td>4.0.4 - Ubuntu 18.04 LTS</td>
</tr>
<tr>
<td>NVIDIA Jarvis Framework</td>
<td>EA v0.2</td>
</tr>
<tr>
<td>NVIDIA NeMo</td>
<td>nvcr.io/nvidia/nemo:v0.10</td>
</tr>
<tr>
<td>Docker container platform</td>
<td>18.06.1-ce [e68fc7a]</td>
</tr>
</tbody>
</table>

Next: Build a Virtual Assistant Using Jarvis, Cloud Sync, and NeMo Overview
Overview

This section provides detail on the implementation of the virtual retail assistant.

Next: Jarvis Deployment

Jarvis Deployment

You can sign up for J**arvis Early Access program** to gain access to Jarvis containers on NVIDIA GPU Cloud (NGC). After receiving credentials from NVIDIA, you can deploy Jarvis using the following steps:

1. Sign-on to NGC.
2. Set your organization on NGC: ea-2-jarvis.
3. Locate Jarvis EA v0.2 assets: Jarvis containers are in Private Registry > Organization Containers.
4. Select Jarvis: navigate to Model Scripts and click Jarvis Quick Start
5. Verify that all assets are working properly.
6. Find the documentation to build your own applications: PDFs can be found in Model Scripts > Jarvis Documentation > File Browser.

Next: Customize States and Flows for Retail Use Case

Customize States and Flows for Retail Use Case

You can customize States and Flows of Dialog Manager for your specific use cases. In our retail example, we have the following four yaml files to direct the conversation according to different intents.

See the following list of file names and description of each file:

- main_flow.yml: Defines the main conversation flows and states and directs the flow to the other three yaml files when necessary.
- retail_flow.yml: Contains states related to retail or points-of-interest questions. The system either provides the information of the nearest store, or the price of a given item.
- weather_flow.yml: Contains states related to weather questions. If the location cannot be determined, the system asks a follow up question to clarify.
- error_flow.yml: Handles cases where user intents do not fall into the above three yaml files. After displaying an error message, the system re-routes back to accepting user questions. The following sections contain the detailed definitions for these yaml files.

**main_flow.yml**

```yaml
name: JarvisRetail
intent_transitions:
  jarvis_error: error
```
price_check: retail_price_check
inventory_check: retail_inventory_check
store_location: retail_store_location
weather.weather: weather
weather.temperature: temperature
weather.sunny: sunny
weather.cloudy: cloudy
weather.snow: snow
weather.rainfall: rain
weather.snow_yes_no: snowfall
weather.rainfall_yes_no: rainfall
weather.temperature_yes_no: tempyesno
weather.humidity: humidity
weather.humidity_yes_no: humidity
navigation.startnavigationpoi: retail # Transitions should be context and slot based. Redirecting for now.
navigation.geteta: retail
navigation.showdirection: retail
navigation.showmappoi: idk_what_you_talkin_about
nomatch.none: idk_what_you_talkin_about
states:
  init:
    type: message_text
    properties:
      text: "Hi, welcome to NARA retail and weather service. How can I help you?"
  input_intent:
    type: input_context
    properties:
      nlp_type: jarvis
      entities:
        intent: dontcare
# This state is executed if the intent was not understood
dont_get_the_intent:
  type: message_text_random
  properties:
    responses:
    - "Sorry I didn't get that! Please come again."
    - "I beg your pardon! Say that again?"
    - "Are we talking about weather? What would you like to know?"
    - "Sorry I know only about the weather"
    - "You can ask me about the weather, the rainfall, the temperature, I don't know much more"
  delay: 0
  transitions:
    next_state: input_intent
idk_what_you_talkin_about:
  type: message_text_random
  properties:
    responses:
    - "Sorry I didn't get that! Please come again."
    - "I beg your pardon! Say that again?"
    - "Are we talking about retail or weather? What would you like to know?"
    - "Sorry I know only about retail and the weather"
    - "You can ask me about retail information or the weather, the rainfall, the temperature. I don't know much more."
  delay: 0
  transitions:
    next_state: input_intent
error:
  type: change_context
  properties:
    update_keys:
      intent: 'error'
  transitions:
    flow: error_flow
retail_inventory_check:
  type: change_context
  properties:
    update_keys:
      intent: 'retail_inventory_check'
  transitions:
    flow: retail_flow
retail_price_check:
  type: change_context
  properties:
    update_keys:
      intent: 'check_item_price'
  transitions:
    flow: retail_flow
retail_store_location:
  type: change_context
  properties:
    update_keys:
      intent: 'find_the_store'
  transitions:
    flow: retail_flow
weather:
  type: change_context
  properties:
    update_keys:
intent: 'weather'
transitions:
    flow: weather_flow

temperature:
    type: change_context
    properties:
        update_keys:
            intent: 'temperature'
    transitions:
        flow: weather_flow

rainfall:
    type: change_context
    properties:
        update_keys:
            intent: 'rainfall'
    transitions:
        flow: weather_flow

sunny:
    type: change_context
    properties:
        update_keys:
            intent: 'sunny'
    transitions:
        flow: weather_flow

cloudy:
    type: change_context
    properties:
        update_keys:
            intent: 'cloudy'
    transitions:
        flow: weather_flow

snow:
    type: change_context
    properties:
        update_keys:
            intent: 'snow'
    transitions:
        flow: weather_flow

rain:
    type: change_context
    properties:
        update_keys:
            intent: 'rain'
    transitions:
        flow: weather_flow

snowfall:
type: change_context
properties:
  update_keys:
    intent: 'snowfall'
transitions:
  flow: weather_flow

tempyesno:
  type: change_context
properties:
  update_keys:
    intent: 'tempyesno'
transitions:
  flow: weather_flow

humidity:
  type: change_context
properties:
  update_keys:
    intent: 'humidity'
transitions:
  flow: weather_flow

end_state:
  type: reset
transitions:
  next_state: init

---

retail_flow.yml

name: retail_flow
states:
  store_location:
    type: conditional_exists
properties:
  key: '{{location}}'
transitions:
  exists: retail_state
  notexists: ask_retail_location
retail_state:
  type: Retail
properties:
transitions:
  next_state: output_retail
output_retail:
  type: message_text
properties:
  text: '{{retail_status}}'
transitions:
  next_state: input_intent
ask_retail_location:
  type: message_text
  properties:
    text: "For which location? I can find the closest store near you."
  transitions:
    next_state: input_retail_location
input_retail_location:
  type: input_user
  properties:
    nlp_type: jarvis
    entities:
      slot: location
      require_match: true
  transitions:
    match: retail_state
    notmatch: check_retail_jarvis_error
output_retail_acknowledge:
  type: message_text_random
  properties:
    responses:
      - 'ok in {{location}}'
      - 'the store in {{location}}'
      - 'I always wanted to shop in {{location}}'
    delay: 0
  transitions:
    next_state: retail_state
output_retail_notlocation:
  type: message_text
  properties:
    text: "I did not understand the location. Can you please repeat?"
  transitions:
    next_state: input_intent
check_retail_jarvis_error:
  type: conditional_exists
  properties:
    key: '{{jarvis_error}}'
  transitions:
    exists: show_retail_jarvis_api_error
    notexists: output_retail_notlocation
show_retail_jarvis_api_error:
  type: message_text
  properties:
    text: "I am having troubled understanding right now. Come again on that?"
transitions:
    next_state: input_intent

weather_flow.yml

name: weather_flow
states:
    check_weather_location:
        type: conditional_exists
        properties:
            key: '{{location}}'
        transitions:
            exists: weather_state
            notexists: ask_weather_location
    weather_state:
        type: Weather
        properties:
        transitions:
            next_state: output_weather
    output_weather:
        type: message_text
        properties:
            text: '{{weather_status}}'
        transitions:
            next_state: input_intent
    ask_weather_location:
        type: message_text
        properties:
            text: "For which location?"
        transitions:
            next_state: input_weather_location
    input_weather_location:
        type: input_user
        properties:
            nlp_type: jarvis
            entities:
                slot: location
                require_match: true
        transitions:
            match: weather_state
            notmatch: check_jarvis_error
    output_weather_acknowledge:
        type: message_text_random
        properties:
        responses:
- 'ok in {{location}}'
- 'the weather in {{location}}'
- 'I always wanted to go in {{location}}'

delay: 0
transitions:
  next_state: weather_state
output_weather_notlocation:
  type: message_text
  properties:
    text: "I did not understand the location, can you please repeat?"
transitions:
  next_state: input_intent
check_jarvis_error:
  type: conditional_exists
  properties:
    key: '{{jarvis_error}}'
transitions:
  exists: show_jarvis_api_error
  notexists: output_weather_notlocation
show_jarvis_api_error:
  type: message_text
  properties:
    text: "I am having troubled understanding right now. Come again on that, else check jarvis services?"
transitions:
  next_state: input_intent

error_flow.yml
name: error_flow
states:
  error_state:
    type: message_text_random
    properties:
      responses:
      - "Sorry I didn't get that!"
      - "Are we talking about retail or weather? What would you like to know?"
      - "Sorry I know only about retail information or the weather"
      - "You can ask me about retail information or the weather, the rainfall, the temperature. I don't know much more"
      - "Let's talk about retail or the weather!"
    delay: 0
  transitions:
    next_state: input_intent

Next: Connect to Third-Party APIs as Fulfillment Engine

**Connect to Third-Party APIs as Fulfillment Engine**

We connected the following third-party APIs as a Fulfillment Engine to answer questions:

- **WeatherStack API**: returns weather, temperature, rainfall, and snow in a given location.
- **Yelp Fusion API**: returns the nearest store information in a given location.
- **eBay Python SDK**: returns the price of a given item.

Next: NetApp Retail Assistant Demonstration

**NetApp Retail Assistant Demonstration**

We recorded a demonstration video of NetApp Retail Assistant (NARA). Click [this link](#) to open the following figure and play the video demonstration.
Use NetApp Cloud Sync to Archive Conversation History

By dumping conversation history into a CSV file once a day, we can then leverage Cloud Sync to download the log files into local storage. The following figure shows the architecture of having Jarvis deployed on-premises and in public clouds, while using Cloud Sync to send conversation history for NeMo training. Details of NeMo training can be found in the section Expand Intent Models Using NeMo Training.
Next: Expand Intent Models Using NeMo Training
Expand Intent Models Using NeMo Training

NVIDIA NeMo is a toolkit built by NVIDIA for creating conversational AI applications. This toolkit includes collections of pre-trained modules for ASR, NLP, and TTS, enabling researchers and data scientists to easily compose complex neural network architectures and put more focus on designing their own applications.

As shown in the previous example, NARA can only handle a limited type of question. This is because the pre-trained NLP model only trains on these types of questions. If we want to enable NARA to handle a broader range of questions, we need to retrain it with our own datasets. Thus, here, we demonstrate how we can use NeMo to extend the NLP model to satisfy the requirements. We start by converting the log collected from NARA into the format for NeMo, and then train with the dataset to enhance the NLP model.

Model

Our goal is to enable NARA to sort the items based on user preferences. For instance, we might ask NARA to suggest the highest-rated sushi restaurant or might want NARA to look up the jeans with the lowest price. To this end, we use the intent detection and slot filling model provided in NeMo as our training model. This model allows NARA to understand the intent of searching preference.

Data Preparation

To train the model, we collect the dataset for this type of question, and convert it to the NeMo format. Here, we listed the files we use to train the model.

dict.intents.csv

This file lists all the intents we want the NeMo to understand. Here, we have two primary intents and one intent only used to categorize the questions that do not fit into any of the primary intents.

```
price_check
find_the_store
unknown
```

dict.slots.csv

This file lists all the slots we can label on our training questions.

```
B-store.type
B-store.name
B-store.status
B-store.hour.start
B-store.hour.end
B-store.hour.day
B-item.type
B-item.name
B-item.color
B-item.size
B-item.quantity
B-location
```
This is the main training dataset. Each line starts with the question following the intent category listing in the file dict.intent.csv. The label is enumerated starting from zero.
Train the Model

docker pull nvcr.io/nvidia/nemo:v0.10

We then use the following command to launch the container. In this command, we limit the container to use a single GPU (GPU ID = 1) since this is a lightweight training exercise. We also map our local workspace /workspace/nemo/ to the folder inside container /nemo.

```
NV_GPU='1' docker run --runtime=nvidia -it --shm-size=16g --network=host --ulimit memlock=-1 --ulimit stack=67108864 -v /workspace/nemo:/nemo --rm nvcr.io/nvidia/nemo:v0.10
```

Inside the container, if we want to start from the original pre-trained BERT model, we can use the following command to start the training procedure. data_dir is the argument to set up the path of the training data. work_dir allows you to configure where you want to store the checkpoint files.

```
cd examples/nlp/intent_detection_slot_tagging/
python joint_intent_slot_with_bert.py -data_dir /nemo/training_data -work_dir /nemo/log
```

If we have new training datasets and want to improve the previous model, we can use the following command to continue from the point we stopped. checkpoint_dir takes the path to the previous checkpoints folder.

```
cd examples/nlp/intent_detection_slot_tagging/
python joint_intent_slot_infer.py -data_dir /nemo/training_data -checkpoint_dir /nemo/log/2020-05-04_18-34-20/checkpoints -eval_file_prefix test
```

Inference the Model

We need to validate the performance of the trained model after a certain number of epochs. The following command allows us to test the query one-by-one. For instance, in this command, we want to check if our
model can properly identify the intention of the query *where can I get the best pasta*.

```bash
cd examples/nlp/intent_detection_slot_tagging/
python joint_intent_slot_infer_b1.py \
--checkpoint_dir /nemo/log/2020-05-29_23-50-58/checkpoints/ \
--query "where can i get the best pasta" \
--data_dir /nemo/training_data/ \
--num_epochs=50
```

Then, the following is the output from the inference. In the output, we can see that our trained model can properly predict the intention *find_the_store*, and return the keywords we are interested in. With these keywords, we enable the NARA to search for what users want and do a more precise search.

```
[Nemo I 2020-05-30 00:06:54 actions:728] Evaluating batch 0 out of 1
[Nemo I 2020-05-30 00:06:55 inference_utils:34] Query: where can i get the best pasta
[Nemo I 2020-05-30 00:06:55 inference_utils:36] Predicted intent:       1
find_the_store
[Nemo I 2020-05-30 00:06:55 inference_utils:50] where   B-interrogative.location
[Nemo I 2020-05-30 00:06:55 inference_utils:50] can     O
[Nemo I 2020-05-30 00:06:55 inference_utils:50] i       O
[Nemo I 2020-05-30 00:06:55 inference_utils:50] get     B-verb
[Nemo I 2020-05-30 00:06:55 inference_utils:50] the     B-article
[Nemo I 2020-05-30 00:06:55 inference_utils:50] best    B-rating.high
[Nemo I 2020-05-30 00:06:55 inference_utils:50] pasta   B-item.type
```

Next: Conclusion

**Conclusion**

A true conversational AI system engages in human-like dialogue, understands context, and provides intelligent responses. Such AI models are often huge and highly complex. With NVIDIA GPUs and NetApp storage, massive, state-of-the-art language models can be trained and optimized to run inference rapidly. This is a major stride towards ending the trade-off between an AI model that is fast versus one that is large and complex. GPU-optimized language understanding models can be integrated into AI applications for industries such as healthcare, retail, and financial services, powering advanced digital voice assistants in smart speakers and customer service lines. These high-quality conversational AI systems allow businesses across verticals to provide previously unattainable personalized services when engaging with customers.

Jarvis enables the deployment of use cases such as virtual assistants, digital avatars, multimodal sensor fusion (CV fused with ASR/NLP/TTS), or any ASR/NLP/TTS/CV stand-alone use case, such as transcription. We built a virtual retail assistant that can answer questions regarding weather, points-of-interest, and inventory pricing. We also demonstrated how to improve the natural language understanding capabilities of the conversational AI system by archiving conversation history using Cloud Sync and training NeMo models on new data.
Acknowledgments

The authors gratefully acknowledge the contributions that were made to this white paper by our esteemed colleagues from NVIDIA: Davide Onofrio, Alex Qi, Sicong Ji, Marty Jain, and Robert Sohigian. The authors would also like to acknowledge the contributions of key NetApp team members: Santosh Rao, David Arnette, Michael Oglesby, Brent Davis, Andy Sayare, Erik Mulder, and Mike McNamara.

Our sincere appreciation and thanks go to all these individuals, who provided insight and expertise that greatly assisted in the creation of this paper.

Where to Find Additional Information

To learn more about the information that is described in this document, see the following resources:

• NVIDIA DGX Station, V100 GPU, GPU Cloud
  ◦ NVIDIA DGX Station
  ◦ NVIDIA V100 Tensor Core GPU
  ◦ NVIDIA NGC

• NVIDIA Jarvis Multimodal Framework
  ◦ NVIDIA Jarvis
    https://developer.nvidia.com/nvidia-jarvis
  ◦ NVIDIA Jarvis Early Access
    https://developer.nvidia.com/nvidia-jarvis-early-access

• NVIDIA NeMo
  ◦ NVIDIA NeMo
    https://developer.nvidia.com/nvidia-nemo
  ◦ Developer Guide
    https://nvidia.github.io/NeMo/

• NetApp AFF systems
  ◦ NetApp AFF A-Series Datasheet
  ◦ NetApp Flash Advantage for All Flash FAS
  ◦ ONTAP 9 Information Library
• NetApp ONTAP FlexGroup Volumes technical report

• NetApp ONTAP AI
  ◦ ONTAP AI with DGX-1 and Cisco Networking Design Guide
  ◦ ONTAP AI with DGX-1 and Cisco Networking Deployment Guide
  ◦ ONTAP AI with DGX-1 and Mellanox Networking Design Guide
  ◦ ONTAP AI with DGX-2 Design Guide