Plotting alignment data

In [15]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import json
import csv
import pandas as pd
import utils.db_utils as db
import utils.plot_utils as plot
import utils.file_utils as file

targetLang = 'en'
bibleType = 'en_ult'
dbPath = f'./data/{bibleType}_alignments.sqlite'

connection = db.initAlignmentDB(dbPath)
```

```
Connection to SQLite DB successful
```

In [16]:
```
# get alignments for tW keyterms

minAlignments = 100
termsPath = './data/kt_en_NT_lemmas.json'
remove = ['ὁ']
lemmasList = db.getFilteredLemmas(termsPath, minAlignments, remove)

# find all alignments for this lemma

alignmentsForWord = db.getAlignmentsForOriginalWords(connection, lemmasList, searchLemma = True)

# filter by number of alignments for word
remove = ['ὁ']
filteredAlignmentsForWord = db.getFilteredAlignmentsForWord(alignmentsForWord, minAlignments, remove)
```

```
'./data/kt_en_NT_lemmas.json' has count: 701
filtered count: 49
```

In [17]:
```
# find all alignments for this original word

# word = 'θεός' # found 69
# word = 'θεὸς' # found 239
# word = 'θεοῦ'  # found 712
# origAlignments = getDataFrameForOriginalWords(connection, word, searchLemma = False)
# origAlignments
```

Analysis of alignments for keyterms in the en_ult:

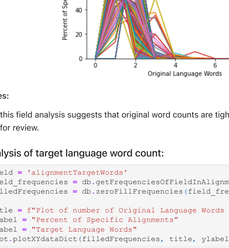
Frequency of alignments:

****Note that each line on the graphs below represents an alignment for a specific word. For example we have separate lines for 'Θεός', 'Θεοῦ', or 'Θεὸν' even though they have the same lemma. It made sense to group the alignments this way since aligners are likely to choose different target language words based on morphology of the word.*

In [18]:
```
frequenciesOfAlignments = db.getFrequenciesOfFieldInAlignments(filteredAlignmentsForWord, 'alignmentTxt')

title = f"Plot of Variability of Specific Alignments in tW KeyTerms"
ylabel = "Percent of Specific Alignments"
xlimit = [0, 10]
plot.plotFrequencies(frequenciesOfAlignments, title, ylabel, showXValues=False, xlimit=xlimit)
```

Plot of Variability of Specific Alignments in tW KeyTerms



Analysis:

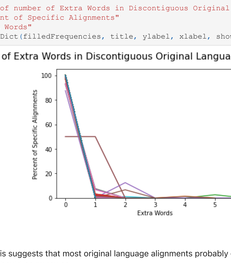
Analysis of numerical metrics:

Analysis of original language word count:

In [19]:
```
field = 'alignmentOrigWords'
field_frequencies = db.getFrequenciesOfFieldInAlignments(alignmentsForWord, field, sortIndex = True)
filledFrequencies = db.zeroFillFrequencies(field_frequencies)

title = f"Plot of number of Original Language Words in Specific Alignments in tW KeyTerms"
ylabel = "Percent of Specific Alignments"
xlabel = "Original Language Words"
plot.plotXYdataDict(filledFrequencies, title, ylabel, xlabel, showXValues=True)
```

Plot of number of Original Language Words in Specific Alignments in tW KeyTerms



Notes:

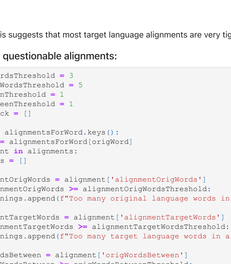
- this field analysis suggests that original word counts are tight - a threshold word count of 3 probably good for English to flag for review.

Analysis of target language word count:

In [20]:
```
field = 'alignmentTargetWords'
field_frequencies = db.getFrequenciesOfFieldInAlignments(alignmentsForWord, field, sortIndex = True)
filledFrequencies = db.zeroFillFrequencies(field_frequencies)

title = f"Plot of number of Original Language Words in Specific Alignments in tW KeyTerms"
ylabel = "Percent of Specific Alignments"
xlabel = "Target Language Words"
plot.plotXYdataDict(filledFrequencies, title, ylabel, xlabel, showXValues=True)
```

Plot of number of Original Language Words in Specific Alignments in tW KeyTerms



Notes:

- this field analysis suggests that a threshold word count of 3 probably good for English to flag for review.

Analysis of count of extra unaligned words between aligned original language words:

In [21]:
```
field = 'origWordsBetween'
field_frequencies = db.getFrequenciesOfFieldInAlignments(alignmentsForWord, field, sortIndex = True)
filledFrequencies = db.zeroFillFrequencies(field_frequencies)

title = f"Plot of number of Extra Words in Discontiguous Original Language Alignments in tW KeyTerms"
ylabel = "Percent of Specific Alignments"
xlabel = "Extra Words"
plot.plotXYdataDict(filledFrequencies, title, ylabel, xlabel, showXValues=True)
```

Plot of number of Extra Words in Discontiguous Original Language Alignments in tW KeyTerms

Notes:

- this field analysis suggests that most original language alignments probably good. Probably the cases of a word between aligned words should be reviewed.

Analysis of count of extra unaligned words between aligned target language words:

In [22]:
```
field = 'targetWordsBetween'
field_frequencies = db.getFrequenciesOfFieldInAlignments(alignmentsForWord, field, sortIndex = True)
filledFrequencies = db.zeroFillFrequencies(field_frequencies)

title = f"Plot of number of Extra Words in Discontiguous Target Language Alignments in tW KeyTerms"
ylabel = "Percent of Specific Alignments"
xlabel = "Extra Words"
plot.plotXYdataDict(filledFrequencies, title, ylabel, xlabel, showXValues=True)
```

Plot of number of Extra Words in Discontiguous Target Language Alignments in tW KeyTerms

Notes:

- this field analysis suggests that most target language alignments are very tight.

Generate CSV of questionable alignments:

In [23]:
```
alignmentOrigWordsThreshold = 3
alignmentTargetWordsThreshold = 5
origWordsBetweenThreshold = 1
targetWordsBetweenThreshold = 1
alignmentsToCheck = []

for origWord in alignmentsForWord.keys():
    alignments = alignmentsForWord[origWord]
    for alignment in alignments:
        warnings = []

        alignmentOrigWords = alignment['alignmentOrigWords']
        if alignmentOrigWords >= alignmentOrigWordsThreshold:
            warnings.append(f"Too many original language words in alignment: {alignmentOrigWords}, threshold

        alignmentTargetWords = alignment['alignmentTargetWords']
        if alignmentTargetWords >= alignmentTargetWordsThreshold:
            warnings.append(f"Too many target language words in alignment: {alignmentTargetWords}, threshold

        origWordsBetween = alignment['origWordsBetween']
        if origWordsBetween >= origWordsBetweenThreshold:
            warnings.append(f"Discontiguous original language alignment, extra words: {origWordsBetween}, th

        targetWordsBetween = alignment['targetWordsBetween']
        if targetWordsBetween >= targetWordsBetweenThreshold:
            warnings.append(f"Discontiguous target language alignment, extra words: {targetWordsBetween}, th

        if len(warnings):
            alignment['warnings'] = json.dumps(warnings, ensure_ascii = False)
            alignmentsToCheck.append(alignment)

basePath = './data/kt_en_NT_warnings'
jsonPath = basePath + '.json'
file.writeJsonFile(jsonPath, alignmentsToCheck)

df = pd.DataFrame(alignmentsToCheck)
csvPath = basePath + '.csv'
warningData = df.drop(columns=["id", "origSpan", "targetSpan"]).sort_values(by=["book_id", "chapter", "verse
warningData.to_csv(path_or_buf=csvPath, index=False, header=True, quoting=csv.QUOTE_NONNUMERIC)
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js